WIM Automation Module (WAM)
USER'S MANUAL

November, 2015

Contents

1 Purpose and Requirements

WIM Automation Module (= WAM) is a major addition to the tools available for WIM users. It allows automating repetitive or complex tasks by using WAM applications that call functions in the WIM library. For the user, WAM has three components. First, WAM programs with the graphical Windows interface make it easy for every WIM user to perform complex operations on a series of images. Second, users with just a little bit more computer knowledge can use command-line WAM programs and run complex applications on series of images. Third, users with interest in programming can use the WIM library to create their own WAM applications using any of the .NET languages, e.g. C#.

WAM is based on the Microsoft .NET Framework. It is realized as a suite of dynamic link libraries that provide easy access to complex functions. Starting with Windows Vista the Microsoft .NET Framework is included in Windows but on older versions of Windows it was necessary to be installed separately.

The default location for WIM and WAM programs (e.g. *C:\Program Files\Wimsoft* in English (US) settings) is added to the PATH system variable when installing WIM/WAM. Sometimes that fails and you need to manually add it. To do that define a new environmental variable, e.g. WIMSOFT, set it to the location of the location of your WIM/WAM installation and add it to the PATH environment variable. You can do it with *Properties* of *My Computer* (right-click on it), *Advanced - Environment Variables*. For example, in the Spanish versions of Windows the default WIMSOFT path is "*C:\Archivos de programa\Wimsoft*" instead of "*C:\Program Files\Wimsoft*".

You can write your own WAM programs. It is best to use the Visual Studio but you can also use the free command line compilers for C# included in the .NET Framework. You can start by compiling and modifying the sample WAM programs. Source codes of various WAM programs (all in the C# language) are included in the *WAM_samples* folder of the WIM/WAM DVD.

WAM functionality is being extended and new functions are added to it. Therefore we recommend checking out the latest copy of WAM whenever you start a new project.

To compile your own programs you also need to add the location of the .NET Framework libraries to the path. For running the command-line C# compiler you also need to add its location to your PATH variable. For the version 2.0 of the .NET Framework it is "*%windir%\Microsoft.NET\Framework\v2.0.50727*".

2 WAM samples overview

2.1 WAM programs with graphical interface

Program Name	Purpose
wam_series	A GUI program designed to apply various operations to a series of matching images. It has options for input and output formats, cutting a sub-area, median filtering, overlays, converting, remapping, changing colors, etc.
wam_match	Finds matches between in situ measurements and a series of satellite images. Makes X-Y scatter plots of the satellite versus in situ data. Creates a CSV text file with match-up statistics of a DX x DY window image statistics corresponding to in situ measurements.
wam_statist	Calculates statistics of masked areas for a series of images. Using a list of images, an image with masked areas or list of stations, calculate statistics for the series of images. Output is a CSV file ready for importing into MS Excel.

2.2 Command-line WAM programs

2.2.1 Anomalies, change detection and EOF

Program Name	Purpose
wam_annual_max	Finds the annual maximum, minimum, time of the maximum, valid counts for each pixel in a series of matching HDF or netCDF datasets.
wam_annual_timing	Finds parameters of the annual cycle: first day and last day over or below a threshold for each pixel as well as counts of images satisfying these conditions in a series of matching HDF or netCDF datasets.
wam_anomaly	Calculates anomalies relative to the mean of overall, monthly 8-day or 5-day annual cycle. Reads all matching images, calculates and saves the means and counts, then calculates and saves the anomalies relative to the means.
wam_change	Calculates change between 2 images, shows the change, i.e. increase or decrease, as either ratio or difference.

wam_eof	Runs EOF or Principal Component Analysis (PCA) on a series of images. See separate document *Exercises_WAM_EOF.pdf* for examples. Starting with WIM 9.x replaced with *wam_pca*.
wam_pca	Runs Principal Component Analysis (PCA) on a series of images. Similar to *wam_eof* but using a different numerical library. See separate document *Exercises_WAM_EOF.pdf* for examples.
wam_kmcluster	Performs k-means cluster analysis on a series of images using CenterSpace NMath Library.
wam_shape	For each pixel in a series of images, evaluates the normalized shape of the series versus a set of shapes, finds the closest shape using RMS differences.
wam_count	Counts the numbers of pixels larger or smaller than a threshold for a whole series of matching images, saves to a text file (CSV file).
wam_trend	Looks for trends in a set of images using the Sen slope or the Linear regression slope and the significance of the slope for each pixel. Replaces *wam_max_trend*.
wam_variance	Calculates pixel-wise mean, standard deviation and the number of valid pixels for a set of matching filenames.

2.2.2 Convert

Program Name	Purpose
wam_compress_hdf	Compresses datasets (SDS) within HDF files. This can drastically reduce file size without any noticeable difference for the user.
wam_uncompress_hdf	Uncompresses datasets (SDS) within HDF files. This increases file size and is used only because some applications cannot use compressed HDF files.
wam_extract_SDS	Extracts one or more datasets (SDS) from a HDF files with multiple datasets into HDF files of a single dataset. Similar to *wam_disassemble*.
wam_assemble_hdf	Assembles datasets (SDS) from a multiple HDF files into a single HDF file with multiple datasets.
wam_disassemble	Disassembles (breaks up) a HDF file with multiple datasets (SDS) into multiple HDF files with a single dataset each.
wam_hdf2nc	Reads all datasets (SDS) from matching HDF4 files and saves in corresponding netCDF files with extension *.nc*.
wam_nc2hdf	Reads all datasets (SDS) from matching netCDF files and saves in corresponding HDF4 files with extension *.hdf*.
wam_convert_100xbyte	Converts the float32 ice concentration HDF datasets to HDF as

	scaled byte.
wam_convert_amsre	Converts datasets of AMSRE in binary format of the Remote Sensing Systems to HDF4. Can read the following datasets: SST, WSPD, VAPOR, CLOUD, RAIN.
wam_convert_ascat	Converts datasets of ASCAT in netCDF to HDF, transforms Latitude and Longitude from Int32 format to Float32.
wam_convert_avisosla	Converts datasets of AVISO netCDF files to HDF4..
wam_convert_ccmp	Converts CCMP wind data in NetCDF format to HDF4.
wam_convert_cm	Converts CM-SAF surface incoming shortwave radiation data in netCDF format to HDF4, adding projection info.
wam_convert_goes1112	Converts GOES 11-12 SST daily data from binary raster to HDF4 format, adds attributes for compositing and time series analysis.
wam_convert_log	Converts HDF datasets to HDF with log scaling of specified slope and intercept.
wam_convert_dim	Converts selected datasets in MERIS DIM files to HDF4 with longitude and latitude arrays for geo-location. See also ***wam_convert_n1***.
wam_convert_k2c	Converts a selected dataset of temperature in Kelvin to Celsius in SST Pathfinder scaling , saves in HDF4 format.
wam_convert_mld	Converts the Mixed Layer Depth datasets in Float32 format to HDF4 in Int16 format, optionally masks land.
wam_convert_n1	Converts the ENVISAT (including MERIS) N1 files to HDF4 with longitude and latitude arrays for geo-location. See also ***wam_convert_dim***.
wam_convert_ncsst	Converts the global 6-km SST data from GHRSST netCDF files into HDF4 files.
wam_convert_ngsst	Converts the New Generation SST (NGSST) binary datasets to HDF4 while adding attributes.
wam_convert_oisst	Converts the OI (Optimum Interpolation) SST (OISST) binary datasets to HDF4 while adding attributes.
wam_convert_percent	Converts percent (range 0.0-100.0) to fraction in a scaled byte (e.g. 0.0-1.0).
wam_convert_ssmi	Converts raster SSMI binary files of ice concentration to HDF4 with added attributes, scaling and projection
wam_convert_to_lin	Converts float32 HDF4 datasets to byte with linear scaling.
wam_convert_to_power	Converts HDF4 datasets to the power of, e.g. 2 or 3.
wam_gradient	Calculates horizontal or vertical spatial gradients.
wam_mirror	Mirrors over vertical, horizontal or both axes; optionally sets projection.

wam_replace_pixel_values	Replaces all unscaled pixel values with a new unscaled pixel value in all matching HDF4 files.
wam_replace_values	Replaces a range of scaled pixel values with a new scaled pixel value in all matching HDF4 files.
wam_lin_transform	Applies a linear transform Y = A * X + B to all valid pixel values of a series of matching HDF image files.

2.2.3 Operations with bands

Program Name	Purpose
wam_band_ratio	Calculates a series of new Chl-a or CDOM images using a set of *Rrs* datasets with a band ratio algorithm, e.g. OC4 or SPGANT.
wam_ratio_2sets	Calculates a series of ratio images using 2 sets of images.
wam_blend_spgant	Blends Southern Ocean product with standard (global) products according to Kahru & Mitchell (2010).

2.2.4 Compositing

Program Name	Purpose
wam_composite	Create an average composite and a count image of a series of HDF images. Reads HDF files either from a list file or from a matching pattern, saves as HDF files. Checks for start and end times to save composite start and end time attributes.
wam_composite_pairs	Reads a list with 2-columns of filenames, composites each pair
wam_composite_quikscat	Reads all Level-3 QuikSCAT files in the folder, calculates the mean composite of ascending and descending orbits, saves as HDF files. Obsolete, needs to be updated.
wam_rotate	Reads pairs of datasets corresponding to respectively eastward and northward wind components and calculates the components in any direction.

2.2.5 Edge or Front Analysis

Program Name	Purpose
wam_edge	Edge (front) detection on a series of images. Runs SIED edge detection algorithm, finds the frequency of edges on a series of SST images.
wam_edge_accumulate	Accumulates statistics of edge images.

2.2.6 Image Operations

Program Name	Purpose
wam_contours	Automates creating isoline contours of *Examine-Contour Lines* in WIM
wam_mozaic_land_ocean	Makes a mosaic from two RGB images based on the pixel value of a mask image.
wam_overlay	Makes a series of overlaid images using a single overlay or respective overlay files.
wam_reduce	Reduces a series of HDF images. Reads all matching HDF files in a folder, reduces the size by a predefined times using a selected reduction type, saves each as HDF.
wam_reduce_valid	Reduces a dataset from a series of HDF or netCDF files while using only valid pixels; can also extract a certain dataset from a file with multiple datasets.
wam_reduce_fixed	Reduces a series of HDF images to a fixed size (192 x 94) while using the valid range.

2.2.7 Mapping

Program Name	Purpose
wam_remap	Remap a series of images, possibly with external geo-referencing to a target projection. Read all matching HDF files, pick SDS, guess the respective geo-referencing file, map to the target projection, save as HDF.
wam_remap2	Remap a series of images, possibly with external geo-referencing to a target projection. Options for selecting SDS, median filtering, fill-holes filling and selected LUT.
wam_remap_all	Remap all datasets of a series HDF4 or netCDF files.
wam_remap_lla	Remap a series of images with external latitude/longitude array (LLA) to a target projection.
wam_remap_and_overlay	Remap and overlay a series of images. The target projection image is also used as an overlay.
wam_remap_to_kmz	Remaps a series of images to a target projection that must be Linear, overlay the target image and annotate. Output is saved as KMZ for visualization in Google Earth.
wam_remap_regions	Remaps a series of images to multiple target projections.
wam_xy2ll	Calculates latitude and longitude from x, y.
wam_ll2xy	Calculates x, y from latitude and longitude.

2.2.8 Primary Production and Export Flux

Program Name	Purpose
wam_npp	Calculates NPP images using the best matching set of images and various algorithms.
wam_npp_lee	Calculates net primary production (NPP) images using the absorption based algorithm of Lee et al (2010) and output from the QAA model (*wam_qaa_l2* or *wam_qaa_l3*).
wam_npp_list	Calculates net primary production (NPP) images using a list of Chl, PAR and SST images and the Behrenfeld-Falkowski (1997) VGPM model with various options.
wam_npp_point	Calculates NPP statistics for a list of "stations" (longitude, latitude, date, time) and series of images using various algorithms.
wam_ef	Creates datasets of export flux of carbon using the various models and input datasets like NPP, SST and Chl-a.

2.2.9 Statistics

Program Name	Purpose
wam_fill	Fills invalid pixels with the mean of the neighboring valid pixels with options for masking land and ice
wam_fill_intime	Fills invalid pixels with linear interpolation from previous and next image or only the next image
wam_fill_with	Fills invalid pixels with the corresponding values from a another matching (in time) dataset
wam_histogram	Calculates histograms of pixel values for a set of 1 byte-per-pixel images in HDF files, saves the histogram in a CSV file
wam_integrate	Integrate a series of images to calculate, e.g., total global primary production or integrated values for specific areas
wam_map_match	Reads a match-up file generated with *wam_match_l2* or *wam_match_nearest* and the corresponding image files and shows the matchup point on the corresponding image
wam_match_l2	Finds match-ups of selected variables in ocean color Level-2 files to a list of stations. Saves statistics of square pixel areas centered at a station
wam_match_multiband	For a list of stations, finds a set of match-ups from different data files (e.g. Rrs bands) from nearest images in time with valid data
wam_match_nearest	For a list of stations finds satellite match-ups of Level-3 data, i.e. the nearest image in time with sufficient valid data and calculates statistics for a square pixel area centered at the station

wam_pixelwise_match	Finds pixelwise matches between 2 sets of images, optionally plots the scatter plot
wam_pixelwise_match_add	Adds pixelwise matches between 2 sets of images to a CSV file produced in *wam_pixelwise_match*, e.g. of other bands
wam_read_match	Reads match-up file generated with *wam_match_l2* or *wam_match_nearest*, filters subsets, makes plots and calculates statistics and can also calculate new variables
wam_pixel_extract	Extracts pixel (specified by its X and Y coordinate) value from a series of images in HDF4 or netCSD files, saves in a CSV text file.
wam_statist_grid	Calculate statistics for latitude/longitude grid for a series of images. Reads a series of images; calculates statistics for a lat/lon grid, saves statistics.
wam_statist_mask	Command line version of *wam_statist*. Calculate statistics for a single area for a series of images.
wam_statist_sta	Reads a list of stations (specified by longitude, latitude, name) and all matching images; calculates statistics for 3x3 pixel areas centered at the station, saves statistics and all 9 pixel values in a CSV text file.
wam_make_mask	Makes a single rectangular mask image, saves as HDF and PNG.
wam_label_mask	Labels the mask image used in *wam_statist*: writes the sequence number into the middle of each mask, saves as HDF and PNG.
sortmasks sortstas sortgrid	Sort the statistics of masked areas by area instead of by image. Sorts the statistics by mask number of a text file created by *wam_statist*. The output files are sorted and ready for plotting with MS Excel.

2.2.10 Screening Level-3 data according to quality flag

Program Name	**Purpose**
wam_screen_goes1112	Using cloud probability as a quality flag image, screens GOES 11-12 SST data, i.e. keeps only the high-quality pixels while replacing others with zeros.
wam_screen_mask	Screens standard files; sorts into folders depending on valid pixels found in a masked area of interest.
wam_screen_pf	Screens AVHRR Pathfinder version 5 files, keeps only the best quality pixels. Uses both SST and quality images, masks low quality pixels, converts to Byte with SST-Pathfinder scaling.
wam_screen_sst_ocpg	Using a quality flag image, screens, i.e. keeps only the high-quality pixels while replacing others with zeros. Screens Level-3 MODIS Aqua (and Terra) SST data produced by the Ocean

	Color Processing Group, converts to Byte with the SST-Pathfinder scaling.

2.2.11 Processing Level-1B MODIS data

Program Name	Purpose
wam_rgb_modis	Creates co-registered true-color RGB composites from MODIS Level-1B data at 1 km, 500 m or 250 m resolution, does accurate band co-registration, optionally maps to a target projection. See separate document *Exercises_modis_250m.pdf* for details.
zoom_modis_lat_lon	Interpolates 1-km MODIS latitude-longitude array to 250-m resolution. Reads all matching MOD03 or MYD03 files and interpolates the latitude and longitude arrays to 250 m. See *Exercises_modis_250m.pdf* for details.
wam_turbidity	Calculates turbidity index from MODIS bands1 and 2, maps to a target projection. Reads a *crefl*-produced HDF file produced with bands 1 and 2, MOD03 geo-referencing file. See *Exercises_modis_250m.pdf*.
wam_turbidity_aster	Calculates turbidity index from ASTER bands1 and 2, maps to a target projection. See *Exercises_ASTER.pdf*.

2.2.12 Processing Level-2 ocean color and SST data

Program Name	Purpose
wam_l2_map wam_composite_2sensors wam_composie_sensors wam_mosaic_2sets wam_composite_2x wam_composite_intime wam_composite_month wam_composite_year wam_composite_last wam_composite_list wam_composite_running wam_rrs_l2	A suite of programs to map and composite NASA Level-2 ocean color and SST datasets. Starting with the Level-2 data and flags to create daily mapped composites. Daily composites are then used to create higher level composites, e.g. multi-sensor, 5-day, 15-day, monthly, yearly, last N day, running N day or composites specified by a time range or a list of files. *wam_rrs_l2* applies various algorithms to level-2 Rrs data.

2.2.13 Merging Level-3 data

Program Name	Purpose
wam_merge_l3	Merge different Level-3 products (even with different resolution). For example, by merging all-weather but low-resolution microwave SST with high-resolution but clear-weather infrared SST you get merged SST product.

2.2.14 Inherent Optical Properties (IOP)

Program Name	Purpose
wam_qaa_l2	Calculates Inherent Optical Properties (IOPs) such as spectral absorption and backscattering coefficients (e.g. a490, aph440, adg440, bbp490) from Level-2 remote sensing reflectance (*Rrs*) data using the Lee et al QAA algorithm.
wam_qaa_l3	Similar to wam_qaa_l2 but using Level-3 remote sensing reflectance (*Rrs*) data.
wam_qaa_match	Calculates Inherent Optical Properties (IOPs) such as spectral absorption and backscattering coefficients (e.g. a490, aph440, adg440, bbp490) from a CSV file with remote sensing reflectance (*Rrs*) data using the Lee et al QAA algorithm.

2.2.15 Correlation between images and a point time series

Program Name	Purpose
wam_correlation	Calculates a correlation image (with each pixel representing the correlation coefficient) between a series of images and a 1-dimensional time series, e.g. an ENSO index.
wam_correlation_series	Calculates a correlation image (with each pixel representing the correlation coefficient) between two series of images.
wam_corrmatrix	Calculates a correlation matrix between two sets of time series.

2.2.16 Simple utilities

Program Name	Purpose
MakeCircles	Draws circles centered at a given list of points with a give radius. This helps to make masks for calculating statistics in circular areas around a set of points with *wam_statist*.
rcrit	Calculates confidence limits for correlation coefficient with a given number of pairs.
wam_add_attribute	Adds a global string attribute to a set of matching HDF files.
wam_add_name	Renames the SDS in the file with the name of the file. Many standard products use the same SDS name, e.g. *l3m_data*. In many cases it is useful to have the images (datasets) named with a name that clearly identifies the image.
wam_cut	Cuts a specified rectangle image from a series of matching images in HDF4 or netCDF files. See also *test_cut_hdf*.
wam_minmax	Shows the valid range for a set of matching HDF files.
wam_setMinMax	Sets the valid range for a set of matching HDF files.
wam_proj_lin	Adds coefficients of a specified Linear projection to a set of

	HDF files.
wam_show_time	Shows the start year, start day, end year and end day of a set of matching HDF files.

2.2.17 Sample WAM programs for learning and testing

Program Name	Purpose
test_cut_hdf	Shows how to create a series of images of a selected sub-area, and save as JPEGs, GIFs or HDFs. This example shows how to read all matching HDF files in a folder, cut out a predefined sub-area, stretch the color scale, save each in different formats.
test_bandratio	Shows how to calculate a new image using a band ratio algorithm. Reads all matching pairs of image files in HDF (e.g. of 443 and 520 nm bands) in a folder, applies a band ratio algorithm, saves the calculated image as a float HDF, converts to a BYTE image using CHL scaling, stretches colors, saves as a JPEG.
wam_filter	Apply a filter, e.g. Median filter, to a series of images, save the results. Reads all matching HDF files in a folder, cuts out a predefined sub-area, applies the Median filter, and saves each as a JPEG.
wam_attribute	Print the value of an HDF attribute from a series of HDF files. Reads all matching HDF files in a folder, extracts the value of an integer attribute (e.g. “Start Day”), and prints the file name and the attribute value.
test_statist	Calculate statistics of a rectangular area. Reads a specified image as HDF, calculates statistics for a specified rectangular area, prints to the screen and saves in a text file.
wam_rgb	Simple example for creating series of RGB composite images. Reads all matching image files from a folder, calculates RGB images from selected bands.
wam_to_hdf	Converts all matching plain raster images in a directory to HDF while adding attributes. Using filenames as clues for time it tries to add attributes like “Start Year”, “Start Day” that are needed for time series analysis.

3 Examples of WAM programs

The following are examples of WAM programs. Some of the programs are quite complex and powerful while some are simple examples. The source codes of all are included. You can compile the example programs with the C# compiler or run the precompiled executables.

To build a simple command-line program, *cd* to the directory of your source code and type “c“ followed by the program’s name without the extension. For example, to build a C# program *test.cs*, cd to your WAM directory and type

c test

On success the batch file (*c.bat*) makes an executable test.exe in your WIMSOFT directory. Remember that the executable must be in the same folder as your WIM executables and DLLs. You can now run the program by typing

test

Remember that your WAM executables have to be in the same folder as the WIM executables and DLLs.

3.1 WAM programs with graphical interface

3.1.1 wam_series

Wam_series is a Windows forms based application using WAM for doing multiple operations on a series of images.

A common scenario for WIM users is that there is a set of images that need the same kind of kind of operations. You can perform the operations manually with WIM but doing it on many images becomes time consuming and error prone. For example, you may have a large batch of global images but you want to cut out a small subset of your area of interest, may-be map it to another projection, overlay coastlines and/or station locations, stretch the color scale, annotate with year and date, and save in a number of different formats (GIF, JPEG, PNG, HDF, Lat/Lon/Value). You can do all this and a set of other operations with a few mouse-clicks using *wam_series*. Another important benefit of using *wam_series* (instead of doing it interactively with WIM) is that you can save a record (log) of your operations and can use the same operations and parameter values another time on a different set of images. As an odd remnant of an early version, *wam_series* can also calculate statistics of a selected rectangle for a series of images although it is recommended to use *wam_statist* for this kind of work.

Wam_series has a number of input fields for specifying the sources, outputs and operations. All selected options are stored in the registry just before processing starts, so even if you close *wam_series* or turn off the PC, the next run of *wam_series* will start the options selected during your last run. However, this method only saves the last set of options. When you want to switch between multiple projects you can save the settings of each projects in a separate log file. All the parameter values are saved in a log file and can be read back from the log file when needed again. The following are the available input parameters:

"**From dir**"is the input directory where the input files area. You can just navigate to the folder where your files are and copy and paste the Address bar.

"**File pattern**" is how you specify the files in the input directory. For example, you may first want to process only files from 1999 and give "S1999*.*" as the pattern and later give "S2000*.*" to process 2000 files. Please be careful with matching too many files: you may match files that are not suitable and produce an error.

"**To dir**" is the folder name where your output files will be written, e.g. "C:\WAMout" could be used for output files from *wam_series*.

"**Save Log**" and "**Get Log**" are buttons for saving/reading the all parameters to/from a log file. When you click on "Get Log" a file picker dialog starts. When you click on "Save Log" you need to have typed the full path of the log file in the text box immediately to the right of the button. When you just start experimenting with *wam_series*, you can ignore the log options. The

logs are for simple text files for saving all the parameters of a project so that you can restore them later.

"**Input type**" lets you specify the input type of files (HDF, netCDF, CoastWatch, Img (Byte), L1B). With HDF and netCDF files you can also specify the sequence number of a particular dataset (starting from 0). In HDF4 and netCDF files containing a single image this should be "0".

"**Save as**" lets you specify the output file type as GIF, JPEG, PNG, HDF, HDF with LLA and/or Lat/Lon/Value. If you specify statistics as processing option then no output files other than the statistics result is created.

The following processing options are available:

Median filter with the window **size** (typically 3 x 3 or 5 x 5 pixels) lets you smooth the output image with a median filter of a specified size.

Fill Holes with the window **size** (typically 1 x 1 or 3 x 3 pixels) lets you fill zero-valued pixels (e.g. missing data due to clouds) with the mean value of the non-zero neighboring pixels. As a side-effect it also fills areas that are not supposed to have valid data, e.g. land if you are working with ocean data. In that case you can mask the unwanted modifications with a custom overlay (see the Overlay option below). Remember that you cannot overlay 0-valued pixels as they will be transparent, i.e. the image will retain its current value and not attain the overlay value. A remedy for that case is to use a small non-zero pixel value (e.g. 1) in the overlay and use Color Stretch with the start value above the non-zero pixel value.

Mirror? (Upside down?) is a seldom used option. It may be needed for Level-1 or Level-2 data (i.e. unmapped) from ascending orbits (e.g. Aqua) where the image may appear upside down. WIM will normally try to mirror the image automatically.

Rotate 90 deg option will rotate the output image by 90 degrees. Currently the geo-location parameters are not converted. If you save just the bitmap (e.g. JPEG or PNG) then it does not matter, however, if you save as HDF and later try to read that HDF file then the geo-location will be wrong.

"**Coastlines**" with **Overlay Pixel Value** lets you specify the WIM coastlines file to be used (for example, C:\Wimdev\Maps\coast_inter.b) and the pixel value of the coastline. Typically, you may want to use 255 (white) for the coastlines to be visible on black background or 1 (purple or black) on white background. Please note that the overlay is created for the first image file and is assumed the same for the remaining images. If you are remapping images to a target projection then they will all have the same size and projection. In that case it is more versatile to create an overlay file – including not just coastlines but possibly lat-lon grid, station locations, land painted to a different color, color bars, etc – and then use the Overlay option (see next).

"**Overlay**" lets you specify an overlay file that you have to generate manually with WIM before running *wam_series*. For example, you may want to put a specific latitude, longitude grid, manually edited color scale, locations of certain stations of labels - practically anything on top of all the images. Bathymetry contours may be used in the overlay image. Remember that the overlay needs to be of the same size as the output files and the background (transparent on the overlay image) needs to have pixel value of 0. Remember that if the overlay pixel value is 0, it will be transparent, i.e. the image will retain its current value and not attain the overlay value.

"**Load Palette**" lets you specify a specific palette file (usually **.lut*) that you load for each image. That will change the color palette of the image. For example, when working with anomalies you may want to use a specific palette (e.g. *anomaly.lut* or *anomaly7.lut*).

"**Annotate**" with selectable **X** and **Y** values lets you specify where an annotation consisting of the year and start day of the image will be put. X means the distance in pixels from the left and Y means the distance in pixels from the top. If you use "-1" for any of these, the default values corresponding to the top right corner will be used. The program tries to guess the start year and Julian day from the image name. This works for most SeaWiFS, MODIS Level-3 images, Pathfinder SST images, etc. If it does not work for your images, please let Wimsoft know and we will try to make it work.

"**Cut subimage?**" with latitude values for the North (top box), South (bottom box), longitude values for the East (right box) and West (left box) lets you cut out a rectangular area as the area of interest for either saving or statistics. Remember to use negative longitude for West longitudes and negative latitude for south latitudes. All values are in decimal degrees, e.g. latitude of 0.5 means 0 degrees and 30.0 minutes north. Please note that if you are remapping images to a target projection then you don't need to use "Cut subimage?". If you cut a wrong area, your remapped image will be empty. If the image has no projection then, of course, you cannot locate certain latitudes and longitudes. For images with no projection the "longitude" and "latitude" values will mean just the x and y of the image coordinates.

"**Statistics**" with "**Valid Min, Max**" selects to calculate statistics for either the whole image or a subarea specified by the "Cut subimage" values. The resulting output file of statistics for the whole series of images is saved in a text file *statistics.txt* in the current folder. The Min and Max values for the valid data range need to be specified as invalid values should be excluded from the statistics calculations. For example, in case of typical chlorophyll-a values the Min and Max can be 0.015 and 64.0, respectively. Please note that Statistics excludes most other operations, i.e. if you are calculating statistics then you are not creating images and vice versa.

"**Min, Max Temperature for CoastWatch**" is only used if the input files are of the CoastWatch type and lets you specify the min and max temperature values. From August, 2006 on this option is no longer available from the GUI. It is still possible to change these values by editing the log file.

"**Convert to byte?**" lets you convert multi-byte (e.g. Int16 and Float) images to simple scaled byte images that are easier to manipulate and visualize. The selectable scalings are "**Chlor_a, Logarithmic**", "**SST-Pathfinder**", "**Pixel_Value**", "**Linear**", "**Logarithmic**". With the last 2 you can also set the "**Slope**" and "**Intercept**" fields.

"**Color stretch**" lets you specify the "**Start**" and "**End**" values corresponding to the *View-Set Colors* (color definition) type of color stretch operation in WIM.

"**Remap to**" lets you remap the image to a selected image in a specific projection (and size). The default mapping option is "Inverse Mapping" that usually produces best looking images but may be very slow for large LLA images. The *Forward Mapping* is usually the fastest but may produce gappy images. The *Forward* with *Fill Gaps* starts with the *Forward Mapping* and then uses *Inverse Mapping* for the gaps.

The output window shows short log from the program. "**Start**" and "**Quit**" buttons are for…, you guessed it!

- When processing large images the program may need a lot of RAM and may take a long time to process a large number of images. It is recommended to start with a small subset of images by selecting a file name pattern that includes only one or two images. You may want to process your series of images in more than one batch, e.g. separating the batches by year, date, etc.

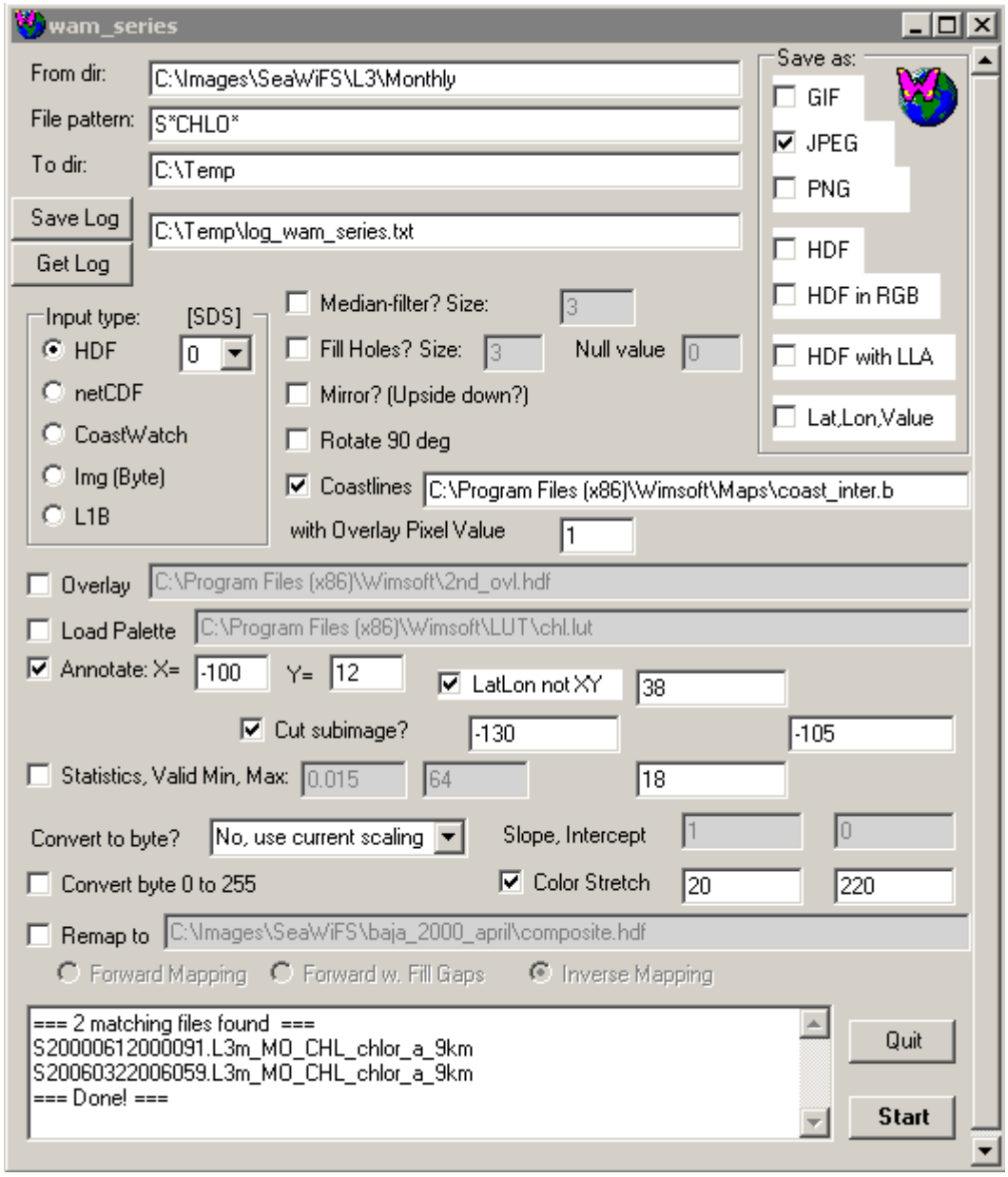
- The recommended coastlines file is *coast_inter.b* (see WIM.pdf = WIM User's Manual for details on GSHHS coastline files).

- When using overlays, all the input files should be of the same size and projection (location). The first overlay is generated dynamically from a selected global coastlines file. This overlay is generated only once, i.e. for the first image. It is then assumed that all the following image files have the same size and projection after processing and before the overlay is being used. It would be possible to remove this restriction and create a new coastlines overlay for each image file but that would make the program slower. Please note the pixel value of the generated overlay. For SeaWiFS Level-3 images you may want to use a pixel value of **1** whereas for Pathfinder SST a good value would be **255**. This is because land pixels on SeaWiFS images are typically white and using pixel value of 255 would typically make a white overlay on white background. In contrast, Pathfinder SST images have land with black pixels (value = 0) and it is appropriate to use white (typically, pixel value 255) as the overlay pixel value.

- The overlay is assumed to be pre-generated with WIM and saved in HDF format. Using a pre-generated overlay can be used, for example, to put bathymetry contours, station locations or other predefined features on the series of images. Bathymetry contours have to be generated in WIM in a 2-step process using *Geo-Bathy Image* first and then applying *Examine-Contour Lines*.

- For HDF files you can select the (0-based) sequence number of the SDS (image) that you want to use from each file. As you know, each HDF file can have many images in the form of SDS (Scientific Datasets). The default is to read the first (0-th) SDS but you can select another SDS. For example, if you want to use "chlor_a" from set of SeaWiFS Level-2 HDF files with the following SDS-s: longitude, latitude, chlor_a, l2_flags, eps_78, K_490, nLw_412,… you need to select "2" for the SDS number. Please note that the sequence of SDS-s in a file is created by WIM and does NOT include the SDS-s that are eliminated by "Settings"-"Special"-"Minimal width of image to be read". For example, let's say that you have set the Minimal width to 200 and the longitude and latitude arrays in a SeaWiFS Level-2 file are only 161 pixels wide. In that case the longitude and latitude are not counted in the sequence of images read by WIM (or WAM). Therefore, the previous sequence of images would become: chlor_a, l2_flags… and instead of "2" you need to select SDS number 0.

The Convert to BYTE function is needed only for multi-byte images (e.g. for int16 and float32 images). The options that are not applicable using the current selections are gray and not available for modification. A screenshot of the default *wam_series* screen is shown below.



3.1.2 wam_match

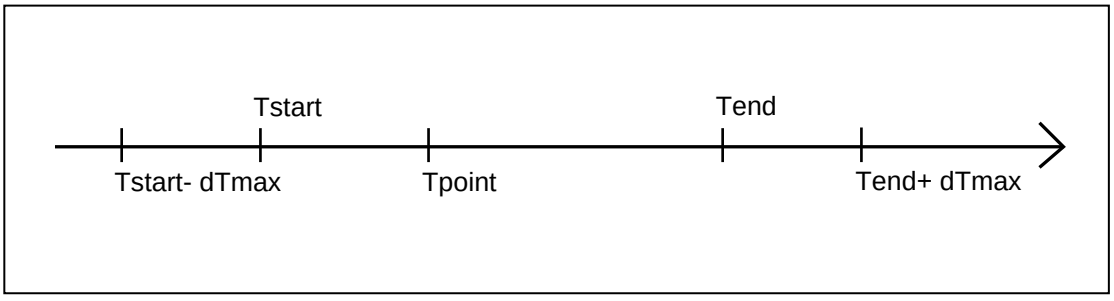
Introduction. The need to understand the relationships between satellite data and ground measurements is fundamental in any application of satellite measurements. The problem is that satellites never measure exactly the same variables as ground measurements. Of course, ground measurements can also use different methods that need to be inter-calibrated and inter-validated. For example, when measuring ocean chlorophyll the ground measurements include variants of HPLC, fluorometry and spectrophotometry. Satellite measurement of chlorophyll is based on radiance spectrum measured at the top of the atmosphere. It is a very complex process to derive, first, the water-leaving radiance and, second, the chlorophyll concentration. There is no one-to-one relationship between the remote and ground methods. Another source of discrepancy is the huge difference in the measurement footprint. The sampling area of a typical ocean color sensor

is 1 km x 1 km whereas ground sampling is usually done with a water bottle with scales of the order of a decimeter (10 cm). A different scenario for doing satellite-ground match-ups is to use satellite data but relate these to totally different kind of ground measurement, for example to compare satellite-detected sea-surface temperature or chlorophyll concentration with the distribution of whales or turtles. The idea of finding matches between satellite and *in situ* data seems simple but in practice the task may easily get overwhelming. Let's say you have a set of measurements that you want to compare with satellite images. You can just move the mouse pointer and by right-clicking in WIM see the geophysical values at a certain location of longitude and latitude. This can be done if you have only a few points and a single image. However, usually we have a whole time series of images and a set of points. The number of all possible pairs of comparisons gets too big too easy. With only 10 points and 10 images we may have to find 10 x 10 = 100 corresponding matching points. We may also want to choose selectively only those images and those points that have the time difference between them below a certain time limit. This means that even with 10 points and 10 images we have to do 100 comparisons of the time difference between a point and an image. We can easily imagine what will happen if we have hundreds of points and hundreds of images: the task just gets overwhelming to be done "manually", i.e. we need a special software program to do that.

Design. *Wam_match* is a complex utility that is based on WAM functions but also uses routines from other libraries (e.g. graphics and data grid). The purpose of *wam_match* is to generate match-up datasets between satellite and *in situ* data. The idea behind this utility is that you have a set of *in situ* data (a set of points) collected from a set of locations in space and time, i.e. specified by longitude, latitude and time. You then use a list of satellite images and try to find matching points between the points and the images by applying several constraints, e.g. that the time interval between the point value and the satellite pass is no larger than *dT* hours and the ground point is inside the satellite image and has valid values (i.e. is not cloud-covered or otherwise invalid). The satellite images can be from a single pass or from a composited image. A composited image is a certain average over the compositing period that is typically 8 days, a month or a year. When making time comparisons between point and satellite data we therefore assume that the point data have a fixed sampling time (*Tpoint*) but the satellite data have a range from start (*Tstart*) to end (*Tend*). We set a maximum time difference *dTmax* and assume that we have a match-up in time between a point and a satellite image if

Tpoint >= (Tstart – dTmax) and dPoint <= (Tend + dTmax).

The following figure illustrates this condition. Point time *Tpoint* can be between *Tstart* and *Tend* but does not have to. It needs to be between *Tstart – dTmax* and *Tend + dTmax*. Typical maximum time difference *dTmax* is 3 or 4 hours when using a single satellite pass but can be relaxed to cover more than 1 day. When using composited images of, say, 8 days, it should be feasible to use *dTmax* of 24 hours or more.



The restriction on space is typically that the point sample is within a certain rectangular window centered at the nearest matching pixel. Typical windows are 3 x 3 or 5 x 5 pixels in size. It is possible to use the value from the single nearest satellite pixel but more often the mean or the median of the valid pixels within the 3 x 3 window are used. This is because the nearest pixel may not be the best match-up with a point sample considering advection by ocean currents during the time interval between the point sample and satellite image.

An important restriction is the validity of the satellite data. Invalid or missing data within an image is caused, e.g. by clouds, and is usually flagged or masked with invalid pixel values. There are two ways of separating valid values from invalid values: using a valid range or a separate image of flags.

- **Valid range with Min and Max** is the easiest way of defining valid pixels. The problem is that the valid range may not obvious and may change for different scalings. Also, when doing match-ups with multiple images in the same file (e.g. *chlor_a*, *nLw412*, *nLw443*, *nLw490*,…) the valid range may be different for the individual images. Also, in some cases we may need to have more details in the classification of pixels based on their quality.
- Using **flags** allows a more detailed characterization of the "quality" of match-ups. For each pixel a set of flags is generated. Each flag indicates a certain condition is confirmed. For example, SeaWiFS Level-2 images have 32 flags each of which can be on or off. These flags include likes of high satellite zenith angle, stray light, high solar zenith angle, navigation error, moderate sun glint, etc. You can pick any combination of these to exclude points. For example, high satellite zenith angle indicates low quality matchup and these points should be excluded from high-quality pixels. Most of the flags, excluding OCEAN flag indicate some problem with the pixel. Using flags to eliminate pixels from match-ups is a versatile method but can only be used if such a flag image is available. *Wam_match* is currently set to use SeaWiFS Level-2 flags and are not suitable for MODIS and other sensors. This may be changed in the next releases of *wam_match*.

A screenshot of the current *wam_match* is shown below. The screen resolution for running *wam_match* should be at least 1024 x 768 pixels. *Wam_match* uses registry to store last parameter values. The following input fields are used.

- "**Use flags**" checkbox indicates if flags are to be used. If selected, a list of 32 flags (currently of the SeaWiFS Level-2 flags) can be selected. The OCEAN should be left unchecked if ocean pixels are sought. You may start with now flags selected and one by one add flags to eliminate pixels. You can start with only a few or no flags checked in order to get many match-ups, and then one-by-one set the flags to see which flags eliminate which match-up points. The eliminated match-ups will be moved to a special "eliminated" set of match-ups and can be examined after saving to a CSV file.
- "**List of Images**" specifies a text file that has a list of image files to be used in the matchup with each image file (full path name, e.g. D:\sat\seawifs\L2\ace0103\V4\S2001105040220.L2_HRBN.sub1.hdf) on a separate line (see a sample list). The image list is easily generated by a command like this:

  dir/b /s S*.hdf > list.txt. The option /s adds the full path to the files (e.g. C:\sat\SeaWiFS\L2\cal0004\). Check out the sample list file *list_mapped.txt*.

  Please note that with image files (e.g. MODIS Level-2, GLI Level-2) having the latitude-longitude arrays (LLA) in a separate file, the list file format is slightly different. After the

first image name, the name of the file with the LLA is listed after a comma. No full path is given for the LLA file, it is assumed to be in the same folder as the primary file.

- "**List of Point data**" specifies is a text file that has a list of ground points (stations) to be used in the matchup. The point list is in the WIM point file format and has tab or comma separated columns of longitude, latitude, date and time followed by station and whatever measurements you have. See included sample point files. The point list is best generated in Excel worksheet and saved as tab or comma separated text file (CSV file). In WIM point files have traditionally had **.pnt* extension but you can also use the **.csv* extension that is more compatible with other programs. The first line in a point file is a header and is not used for data. The format of a point file is the following:

```
Lon   Lat   Date       Time       Cruise     Station    Pt_Value
-117.303   32.958     4/7/2000   22:07 CAL0004    93.26.7    2.11
-117.401   32.930     4/7/2000   23:07 CAL0004    93.28      2.11
```

Any number of lines can follow the header with each point as a separate line. Points can be excluded without deleting the line by preceding the line with the # character. The minimum set of columns is *Longitude*, *Latitude*, *Date*. If *Time* is missing then it is assumed to be 12:00.

Check out the sample point file *match.csv*.

The date has to be in the US format of MM/DD/YYYY or the European format (DD/MM/YYYY). The European format is assumed only if a decimal comma is detected in the *Longitude* value.

- "**Time lag, hr**" specifies the maximum time lag in hours from the start or end time of the image in hours (see a figure above). Please note that in case of composites the interval between the start and end time can be a month (in case of a monthly composite) and the time lag is counted from the start and end.
- "**Windows, DX x DY**" specifies the size of the window in pixels where the matchup statistics is calculated. 3 x 3 pixels is typically used but may be increased to 5 x 5 or even bigger.
- "**Min Valid Pixels**" specifies the minimum number of valid pixels in the window to accept a matchup. For example, in case of 3 x 3 window we may request that at least 5 pixels (out of 9) be valid in order to accept a matchup.
- "**Save results in**" specifies a file where the results will be save in CSV (Comma Separated) format.
- "**Bands**" with individual checkboxes for up to 16 bands (starting from band 0). HDF files (e.g. SeaWiFS Level-2 files) may have many images in the same file and you may select more than one bands for match-ups. If you selected to use flags ("Use flags") then the flags image will be used anyway (if found). The standard mapped images (SMI) in Level-3 have typically a single image and the 0^{th} band should therefore be used. For example, if you want to use "chlor_a" from set of SeaWiFS Level-2 HDF files with the following scientific datasets (SDS): longitude, latitude, chlor_a, l2_flags, eps_78, K_490, nLw_412,… you need to select only band "2". Please note that the sequence of SDS-s in a file is created by WIM and does NOT include those SDS-s that are eliminated by *Settings-Special-Minimal width of image to be read*. You can check the bands with WIM before searching for match-ups.

- "**Valid range, Max, Min**" specifies the minimum and maximum value to be considered valid. Only valid pixels are used in the statistics calculations. It is only used when "Use flags" is not selected.

After the user clicks on the "**Start**" button the search for match-ups begins. A log of output is printed into the text box while the found match-ups (if any!) are put into a *data-grid* and into a graph. In this screenshot example given below 34 matches were found. There are now several ways of tracing the origins of each matchup. You can select variables to be plotted from the point dataset (X-axis) versus variables from the satellite data (Y-axis) as well as the point's "Label" variable using respective up-down controls. The point's "Label" is shown in the "**Point Label**" text box. A "label" is actually any field in the point table (e.g. longitude, latitude, date, time, etc.) and can be selected in the "**Select Point Label**" domain up-down control. When moving the mouse cursor to a particular point you can see the point label in the "Point label" text box. Also, in the data-grid the corresponding row is selected. Vice versa, when a row is selected in the data-grid by selecting it with a mouse click, the corresponding point in the plot will be surrounded by a blue diamond frame. The value is used to represent **missing values** in both the point and satellite data can be chosen. Typical values are 0 or -9.

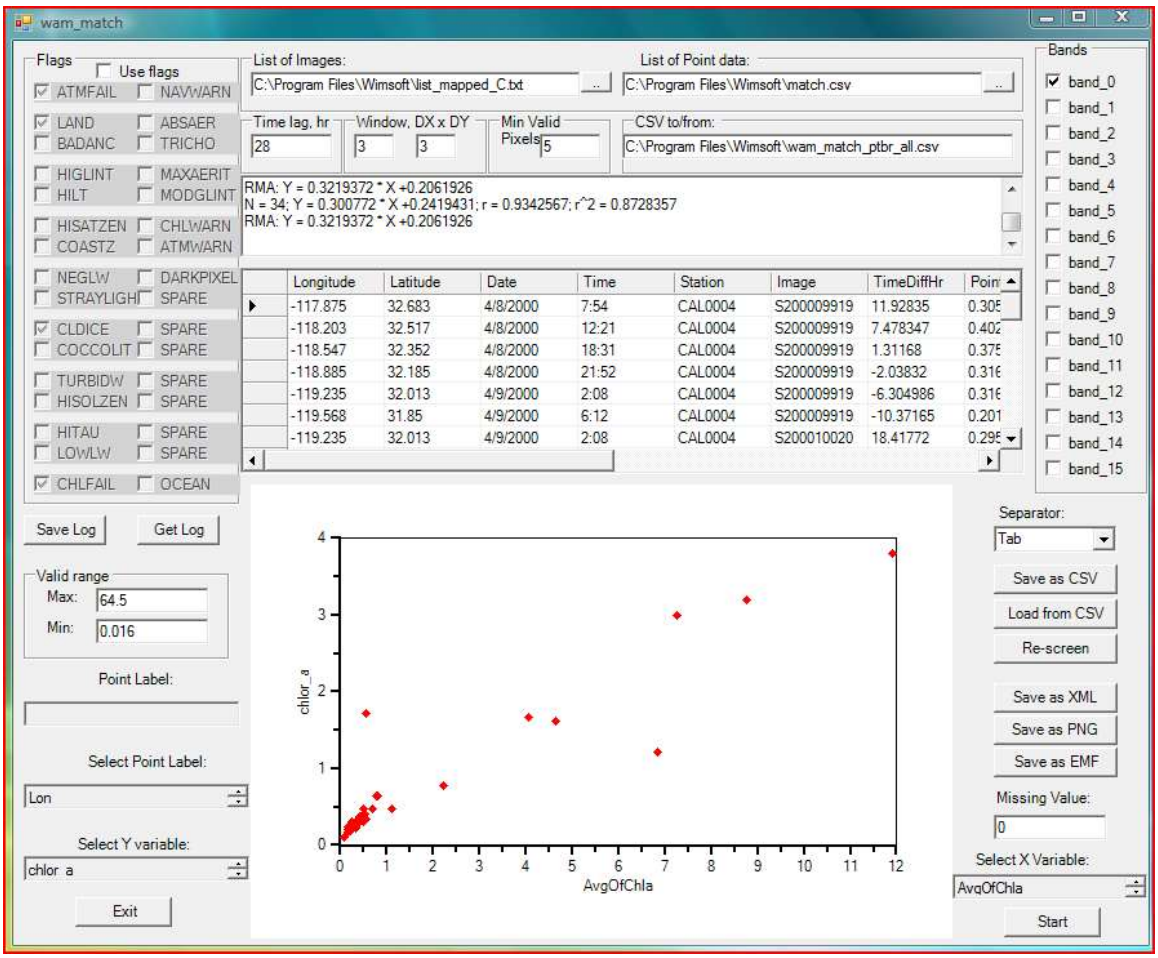
All matching points can be saved in a CSV (comma separated) file than can be loaded into Excel for further analysis. By double-clicking on a point in the graph that point will be eliminated from the set of match-ups shown in the graph and in the data-grid, and placed into a set of "eliminated" matches. Both the retained and eliminated matches will be saved to a CSV (comma separated) file when "**Save as CSV**" button is clicked. Respectively, "**Save as XML**" saves the matches in an XML files but this function is not properly implemented yet. "**Save as PNG**" and "**Save as EMF**" buttons save the graph in the PNG (portable network graphics) and EMF (enhanced windows metafile) formats, respectively.

For running *wam_match* you basically need a set of satellite images, a list of these image files in a list file and a point file with a list of ground samples. Please note the format of these files, especially of the point file and use exactly the same format of representing dates and times. Check out the sample list file *list_mapped_C.txt* and sample point file *match.csv*. These files should be in your WimSoft folder (e.g. *C:\Program Files\WimSoft*). The results will be probably wrong if you use a different format.

- First try to replicate the results of the figure below with the sample files. You need to copy the sample images from the WIM/WAM CD to your hard disk. The sample list file expects them to be in *C:\Program Files\WimSoft\Images\SeaWiFS\baja_2000_april*. You can get this if you just manually copy the *Images* folder from the CD to your *WimSoft* folder. The *Images* folder has a lot more files than are needed for this example but you may delete them later if not needed.
- Fill out all the control boxes exactly as in the figure below, then click *Start*. You should get exactly the same results, i.e. 34 matches. As you see some of the matches look like outliers. If you move your mouse pointer to the point in the scatter plot, the corresponding row in the data grid will highlight. If you double-click on the outlier (or any other match-up point for that matter), the point will be removed from the plot and from the match-up set to the eliminated set of match-up points and the figure will be rescaled. Both sets (match-up points and eliminated match-up points) will be saved to a comma separated (CSV) files if you click Save as CSV. The graphs can be saved as PNG or EMF. Note that in this example only one image (SDS) was present in the image files, also that no flags were used.

- Now you are ready to try to match up your data with satellite images. Just please follow the formats.

Related command line utilities, such as *wam_match_nearest*, *wam_match_multiband* and *wam_match_l2*, do not need a list file of images. Instead, they find the nearest image in time with sufficient number of valid pixels corresponding to a station (defined by longitude, latitude, date and time).



3.1.3 wam_statist

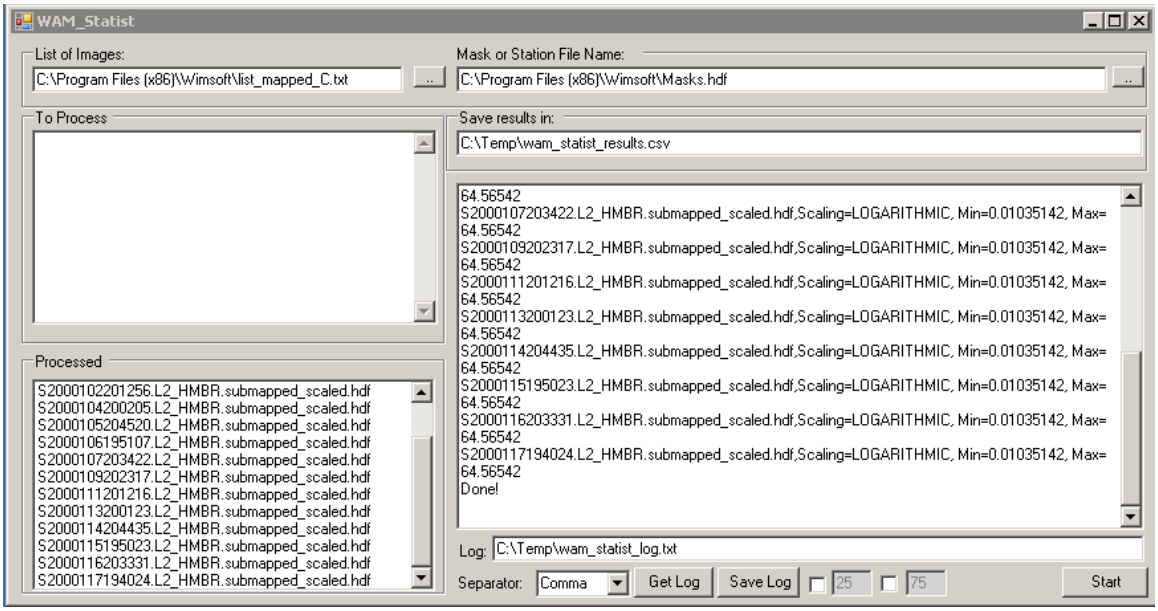
wam_statist is a Windows Forms program that combines the functionality of *wam_statist_sta* and *wam_integ_mask* with a GUI and should be easier to use.

Main idea is to use a **List** image files specifying the images and a **Mask** image, specifying the areas, and calculate statistics for the images over the specified areas. The main input fields are therefore "List of Images" and "Mask or Station File name". You can make your selections either by typing into the text box or picking the file with the file picker button. The image files specified in the "List of Images" are expected to be in HDF4 format. Starting with WIM 6.x NetCDF files can also be used and they are expected to have *.nc* extension.

When specifying output file, pick a directory where you have write permission. Do not try to save into system directories like "`C:\Program Files (x86)`", etc. It is recommended to also save a

new and different **Log** file with each output file. A Log file allows to record the settings used in creating the output statistics file. A good naming practice is to use the same filename for the log, except changing the extension from *.csv* to *.txt*.

Please note that the distinction between **mask** statistics and **station** (point) statistics is based on the filename extension of the "Mask or Station file name" text box. Mask files are assumed to be only with the *.hdf* extension (HDF files) or *.rle* extension (WIM compressed image files) and the text files with station coordinates and names are assumed to have the *.csv* or *.pnt* extension. Please note that the format of the station list assumes at least 3 columns: "*Longitude, Latitude, Station*" or "*Latitude, Longitude, Station*". The default is to have *Longitude* first and *Latitude* second but if names *Longitude* or *Latitude* are found reversed in the header then the sequence is adjusted accordingly. After the 3 columns, i.e. *Longitude, Latitude, Station*, you can have any other columns. When the calculations on an image are finished, the image name is moved from the top box to the bottom box. The output has the usual WIM statistics. When doing statistics for stations (i.e. points), the statistics is done for the 3 x 3 pixel window centered at the nearest pixel and the individual values of the 9 pixels are added to the statistics. The default separator in the output file is "comma" that works fine with the English (US) system of formatting. In many other "culture" settings comma is used as a decimal separator (like decimal point in the US) and therefore cannot be used as a separator between numbers. "Tab", "Space" or "Semicolon" are the other options. "Semicolon" should work in most cultures. In addition to the regular statistics *wam_statist* calculates the *Total* column. For images in the global equal angle projection *Total* is the sum of pixel area times the pixel geophysical value. For example, in case of primary production *Total* is the total primary production in the masked area. For other projections *Total* is just the sum of pixel geophysical values, i.e. it is not an integral over the area.



3.2 Command-line WAM programs

NOTE! The current syntax of these command-line utilities may have changed! You can always check the syntax by typing the name of the program without arguments.

3.2.1 Anomalies, change detection and EOF

wam_annual_max

wam_annual_max vers. 2.01
Usage: wam_annual_max Pattern
Options:
Median=Y where Y is 3, 5,... applies median filter of size Y to all images
Reduce=X where X is 2, 3, 4,... reduces the image size by X times

wam_annual_max scans a series of HDF4 or netCDF files matching *Pattern*, reads the first dataset and for each pixel finds the maximum (*Max*), minimum (*Min*) and number of valid counts (*ValidCounts*). Starting with version 2.x it also records the time of the maximum in year days (*MaxDay*). Previously that was available only with the (now obsolete) *Slow* option. The *Reduce* option reduces the images by an integer factor of times (e.g. 2, 3,..). This is needed if the images are big and there is not enough RAM to run the command.

Examples and more information in *Detection_of_Change.pdf* and *Detection_of_Change_Short.pdf*.

wam_annual_timing

wam_annual_timing vers. 1.13
Usage: wam_annual_timing Pattern +/-/C Threshold
Pattern is the pattern of matching HDF4 or netCDF files.
+ or - means that pixels larger or smaller than the Threshold are counted
C means that we are counting cumulative values larger than the Threshold
Threshold is the threshold value
Options:
After=A considers only days > A
Before=B considers only days < B
Median=Yes smoothes the image with 3x3 median filter
Reduce=R where R is reduce factor, e.g. 2
SaveValidCount=Yes saves images of valid pixel counts
SaveCumulativeSum=Yes saves images of cumulative sum - only with C option
Example to find the first and last days when SST is at least 20 C:
wam_annual_timing C:\SST*.hdf + 20
Note the space between + and 20!
Example to find the first and last days with wind speed at least 3 during a period of days 100-150:
wam_annual_timing C:\Windspeed*.hdf + 3 after=99 before=151

Imagine that you want to find the extents of the ice period (i.e. how many days the area is covered by ice), the start and end of the ice period for each year in a series of daily ice image. *wam_annual_timing* will calculate that for many years with a single command. You can also do the opposite, i.e. calculate the period (count) of no ice in days, the first day and the last day of no ice for each year. Of course, you can also apply the same operations to images other than ice concentration and you will get something like the number of days of Chl-a or SST higher or lower than a threshold, the first and last days of higher/lower than a threshold.

The options allow you to choose the threshold value, the operation of finding pixels that are either higher (+) or lower (-) than the threshold.

You may need to do creative sorting of files if you want more complicated timing indicators. For example, if you want to find the day of the year of freeze-up, e.g. when ice concentration becomes higher than 0.5 and you are in the Northern Hemisphere then you will probably get day = 1 as January 1 already has ice concentration > 0.5. If you are really looking for freeze-up that happens in, say, November-December, then you need to apply this command to files of November-December only. Similarly, in the Southern Hemisphere you may need to sort files if you are looking for the ice retreat happening in, say, October, then January-1 images may already have ice concentration below threshold and you are not getting what you need.

The *Reduce* options are similar to *wam_annual_max* and allow reducing the images and the required memory.

wam_anomaly

```
wam_anomaly vers. 3.40
Usage: wam_anomaly Pattern 1/12/25/46/73
where 1=> annual, 12=> monthly, 25=> 15-day, 46=> 8-day, 73=> 5-day anomalies
Options: SDS=N LUT=PaletteFile Show=true/false Overlay=file Mean=Meanfile
  AnnotateX=X AnnotateY=Y JPG=no
SDS=N means that the Nth SDS is used in the multi-sds file. Default is SDS=0.
LUT=anomaly.lut specifies the LUT file to be used. Default is anomaly5.lut.
If Show=true then invalid pixels are shown in black, otherwise as white, i.e. no anomaly
Overlay=file specifies the image to be overlaid on top of the anomaly images
Mean=filename means that you are using a precalculated Means file
AnnotateX and AnnotateY are, respectively, X and Y in pixels where to put the
  Date annotation (default is the upper right corner)
JPG=N means anomaly images are NOT saved as JPG, default is to save both HDF and JPG
      images.
ReduceX=X means that the JPG anomaly images are reduced reduceX times.
  Default is to reduce automatically if image width > 800 pixels.
Type=Difference or Type=Ratio forces to use either Difference or Ratio anomaly.
Default is to detect the type, e.g. Diff for SST and Ratio for Chla or NPP
MinAnomaly=Min, MaxAnomaly=Max set the anomaly range;
Default is to set the range automatically.
Basic example for calculating monthly anomalies:
  wam_anomaly Data\*.hdf 12
Another example for monthly anomalies:
  wam_anomaly Data\*.hdf 12 LUT=anomaly5.lut Show=true Overlay=Maps\landmask.hdf
```

wam_anomaly is a powerful command-line program that calculates the means and the anomalies from the means. As the "mean" it can assume annual cycle with different intervals. In the monthly mode (argument 12) it calculates twelve monthly means and anomalies from the monthly means. In the annual mode (argument 1) it calculates a single general mean and anomalies from the general mean.

Pattern is a matching filename pattern, for example *S*.hdf*. The second argument specifies the interval of the annual cycle to be used. The mean annual cycle and anomalies from it are calculated for annual (1), monthly (12) , 8-day (46) and 5-day (73) intervals. *PaletteFile* is a specified palette file (e.g. *anomaly5.lut*), *MaskFile* is an optional mask file (see below). As

output, the program generates *Mean.hdf* using the specified interval (e.g. monthly means if option 12 is used as 2nd argument), *ValidCounts.hdf* – the number of valid pixels used in the calculation of the means, and anomaly images for each of the matching image used. Option SDS=N allows to read the Nth dataset in a multi-dataset HDF file. The default is to use the first (0th) dataset.

If the *1* (annual) option is used then you want to find the annual anomalies compared to long-time mean. If the *12* (monthly) option is used then you want to find the monthly means and the monthly anomalies compared to the interannual mean for a particular month.

You can use *wam_series* to generate a series of images for your area of interest and then run *wam_anomaly* to create the anomaly image series. Using it on a series of monthly images and making monthly means makes sense if you have more than one year of monthly data. For SST the anomalies are calculated by subtracting the corresponding monthly mean from each image data. However, you can change the default behavior by using command line options. These options force to use either Difference or Ratio anomaly. Also, you can specify the anomaly range in the command line. For chlorophyll the default anomaly is calculated as the ratio of the image to the corresponding monthly mean. By default the SST anomaly is scaled to the limits of ± 5 degrees C of a byte image, i.e. pixel value 128 corresponds to the mean (i.e. anomaly of 0), pixel value of 254 corresponds to positive anomaly of ~5 C, pixel value of 1 corresponds to negative anomaly of ~5 C. Pixel values 0 and 255 are normally reserved for out of bounds values or no data. For chlorophyll the maximum positive anomaly is 10 times the corresponding monthly mean and the minimum anomaly is 0.1 times the corresponding monthly mean. For visualization of the anomalies you can use a special two-tone palette (*anomaly.lut* or *anomaly7.lut*) that has red for positive and blue for negative anomalies and the saturation shows the intensity of the anomaly. Some of the optional functions in *wam_anomaly* (e.g. *Median filter*, *Fill Holes*) fill in missing values and smooth the images and extend the valid areas, e.g. ocean values over land. To prevent that you can use an Overlay image that has non-zero pixel values over land. All pixels that have nonzero values will be overwritten with the overlay value in the anomaly image.

Please note that when using filenames as arguments (e.g. the Palette File) you cannot use spaces. A space would mean a separation between consecutive arguments. For example, you cannot use a Palette file *C:\Program Files\Wimsoft\anomaly7.lut* as an argument. Instead, you need to copy the palette file to a folder with no spaces in the name, e.g. *C:\sat*.

wam_change

```
wam_change vers. 1.5
Usage: wam_change File1 File2 [AnnotateX AnnotateY LUT]
  AnnotateX and AnnotateY are, respectively, X and Y in pixels
   where to put the
Date annotation
  The default LUT is anomaly5.lut
```

wam_change calculates the change from image in File1 to the image in File2 and shows the difference like an anomaly. With the default color palette increase from Image1 to Image2 is shown in red and decrease from Image1 to Image2 is shown in blue. Depending on the data type it uses either the pixel-wise ratio (Image2/Image1, e.g. for Chl-a) or pixel-wise difference (Image2 – Image1, e.g. for SST and ice concentration). Depending on your data type it may be needed to adapt the source code for your data files to produce the most appropriate scaling.

wam_eof ******* Starting with WIM 9.x replaced with *wam_pca*.

```
wam_eof vers. 1.6
Usage: wam_eof FilePattern [Mask.hdf [DemeaningOption]
  FilePattern is a filename pattern and all matching files will be processed
  Mask.hdf is a byte image in HDF format. Pixels to be included are nonzero, pixels to be
     excluded (e.g. land) are zero
  DemeaningOption should be either 'Pixel' to subtract the corresponding mean pixel value over
     all images, or 'Image' to subtract the image mean from each image, or 'No' for no
     demeaning at all.
  The default is to subtract the 'Pixel' mean.
HDF files like PC_* are excluded as they are assumed to be saved principal compnents from
     previous runs.
```

wam_eof performs EOF (empirical orthogonal function) analysis on a series of images. It is closely related to the Principal Component Analysis. It is typically run after running *wam_anomaly*, i.e. on anomaly images. Please see a separate document *Exercises_WAM_EOF.pdf* for detailed instructions and examples.

wam_pca

```
wam_pca  vers. 1.10
Usage: wam_pca List/Pattern [Mask.hdf [DemeaningOption]
Can use either a List of filenames or a Pattern of matching HDF4/netCDF files
Mask.hdf is a byte image in HDF4 format. Pixels to be included are nonzero,
 pixels to be excluded (e.g. land) are zero.
DemeaningOption should be either 'Pixel' to subtract the corresponding mean
 pixel value over all images,
  or 'Image' to subtract the image mean from each image,
 or 'No' for no demeaning at all.
The default is to subtract the 'Pixel' mean.
HDF files like PC_* are excluded as they are assumed to be saved principal
 components from previous runs.
```

wam_pca performs Principal Component analysis on a series of images. It is similar to *wam_eof* and the EOF analysis except that it uses a different numerical library. It is typically run after running *wam_anomaly*, i.e. on anomaly images. Please see a separate document *Exercises_WAM_EOF.pdf* for detailed instructions and examples.

wam_kmcluster

```
wam_ kmcluster vers. 1.1
Usage: wam_kmcluster FilePattern K [Mask.hdf [DemeaningOption]
  FilePattern is a filename pattern and all matching files will be processed
  K is the number of clusters
  Mask.hdf is a byte image in HDF format. Pixels to be included are nonzero,
 pixels to be excluded (e.g. land) are zero.
  DemeaningOption should be either 'Pixel' to subtract the corresponding mean
 pixel value over all images,
  or 'Image' to subtract the image mean from each image,
 or 'No' for no demeaning at all.
  The default is to subtract the 'Pixel' mean.
HDF files like PC_* are excluded as they are assumed to be saved principal
```

components from previous runs.

wam_kmcluster performs k-means cluster analysis on a series of images using CenterSpace NMath library. The k-means clustering method assigns data points into k groups such that the sum of squares from points to the computed cluster centers is minimized. It implements the Hartigan and Wong algorithm (A K-means clustering algorithm. Applied Statistics 28, 100–108. 1979) with the following steps:

1. For each point, move it to another cluster if that would lower the sum of squares from points to the computed cluster centers.

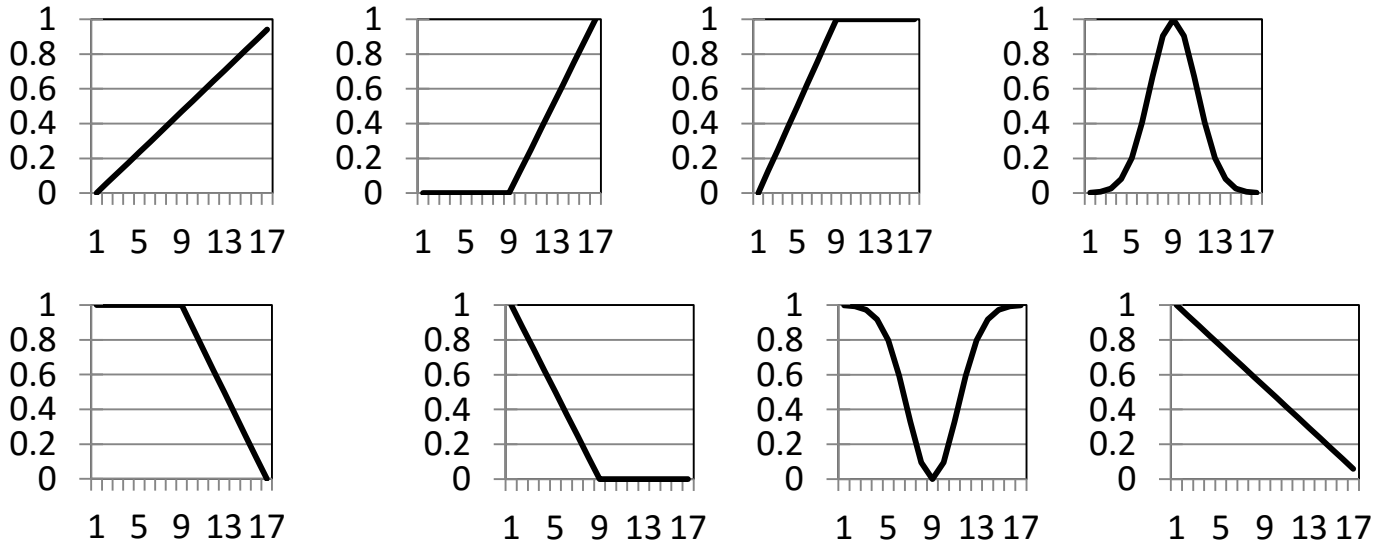
2. If a point is moved, immediately update the cluster centers of the two affected clusters.

3. Repeat until no points are moved, or the specified maximum number of iterations is reached.

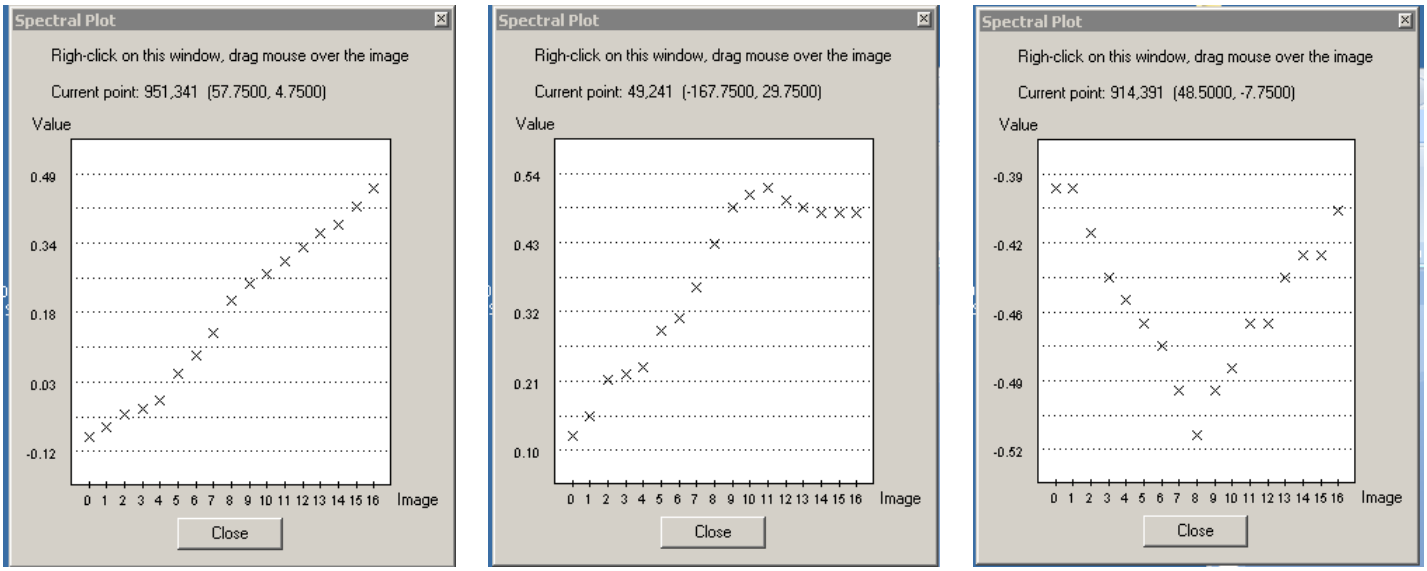
wam_shape

```
wam_shape vers. 1.1
Usage: wam_shape List/Pattern [Mask.hdf][Power][LimitSS=X]
  Can use either a List of filenames or a Pattern of matching filenames
Mask.hdf is a byte image in HDF format. Pixels to be included are nonzero,
 pixels to be excluded (e.g. land) are zero.
Power is an option to save the best shape number as 2^(index - 1)
  This set Value scaling to BitMask and shows the shape names with right-click
LimitSS=X selects X as the sum of squares (SS) below which the selected shape
  is considered acceptable. If SS of the best matching shape is higher than X
  then it is set to 0, i.e. not detected. The default value for LimitSS is 2.
```

wam_shape evaluates the normalized shapes of time series in each pixel relative to a set of shapes. Tere are currently 8 preset shapes but these can be augmented in the future. Te current shapes are: (1) *linear increase*, (2) *plateau to increase*, (3) *increase to plateau*, (4) *normal hump*, (5) *plateau to decrease*, (6) *decrease to plateau*, (7) *normal trough*, (8) *linear decrease* (see fig. below).



The argument is either a text file with a list of HDF filenames or a matching pattern of HDF filenames. The list is preferred as in the list you can manually edit the sequence of files which is important for shapes. For each pixel of the mat6ching images *wam_shape* creates a normalized time series between 0 and 1. You can interactively look at your per-pixel time series with *Examine-Spectral Plot* function in WIM (see fig below). You can see that in each pixel location the shapes are somewhat different. The figure below shows linear increase (left), increase to a plateau (center) and a minimum in the middle (right panel). The example here uses 17 images; therefore the horizontal axis shows values from 1 to 17. These time series will be normalized to the range of 0 to 1 and their shapes will be compared to the preset shapes by calculating the root mean square (RMS) differences with the preset shapes. The best matching shape is selected by the minimum RMS difference. In the example below, the left panel is closest to the shape *linear increase*; the center panel is closest to *increase to plateau*, and the right panel is closest to *normal trough* (upside down normal distribution)



A new multi-SDS HDF4 file is created where the first (0-th) image shows the best matching shape and the following images show the RMS differences with each of the shapes (the lower the value, the best is the match). In order to show the name of the shape when right-clicking on the best shape image, when the *Power* option is selected, *Value Scaling* is set to *BitMask* instead of *Pixel Value* and underlying pixel value is not the shape number but 2 to the power of (*shape number* – 1). For example, without the *Power* option the pixel values of *linear increase*, *plateau to increase*, *increase to plateau*, *normal hump*, *plateau to decrease*, *decrease to plateau*, *normal trough* and *linear decrease* are 1, 2, 3, 4, 5, 6, 7, 8, respectively, whereas with the *Power* option they are 1, 2, 4, 8, 16, 32, 64, 128. When *Value Scaling* is set to *BitMask* (with the *Power* option) you will see the shape names when right-clicking on the image. To see the pixel values you need to change *Value Scaling* to *Pixel Value* in WIM *Settings* or not use the *Power* option.

wam_count

wam_count vers. 1.5
Usage: wam_count Pattern Threshold +/- [Mask][Area][NoTest][Sort]
where Pattern is a matching pattern of HDF4 filenames,
Threshold is the upper or lower threshold value,

+ or - means that pixels larger or smaller than the Threshold are counted.
Mask is an optional HDF4 file that specifies areas of interest with pixel values of 1, 2, ... Up to 255 diferent masks can be used.
Mask must have the same size as all the images.
'Area' option makes it to calculate area in km2 instead of pixels.
Area is calculated correctly only for global standard mapped images (SMI).
'NoTest' option forces to use ALL pixels (Valid or Invalid).
Default is to count only Valid pixels.
'Sort' option sorts images by pixel count, starting from the highest count.
Output is saved in the current directory as a *.csv file.

wam_count is a utility that reads a series of images from matching HDF4 files (only the first dataset is read) and either counts the number of pixels or the total area corresponding to either larger or smaller than a specified threshold. For example, you may be interested to create a time series of the area with temperature over a threshold value (or below a threshold value) or the size of the area with very high Chl-a levels (e.g. corresponding to a red tide). With *wam_count* you can create multiple time series corresponding to certain masked areas with a single command.

The results of *wam_count* are saved in a text file in the CSV file format that is suitable for importing into Excel. Without the *Mask* option the whole image is scanned and the pixels either larger or smaller than a threshold are counted. With the *Mask* option only those pixels that correspond to the masked areas are used. The *Mask* file is a simple byte image read from a HDF file with masked pixels different from 0. Multiple masks can be specified with pixel values 1, 2, …, up to 255. Counts are then generated for each of the masked areas. If the Mask option is used then all images must have the same size equal to the size of the *Mask* image. The *Mask* file is similar to the mask file used in the *wam_statist* utility. Mask files can be created with the *Edit-Draw* functions in WIM (see the manual pages of *wam_statist*). Note that comparisons of double precision numbers can be tricky. If we want to find the counts of pixels with values larger or equal than 10, in order to include values of 10.0 written like 9.9999999999, we have to use the threshold slightly smaller than 10, e.g. 9.99.

Let's say that we want to calculate time series of the area of sea surface with temperature above 28.5C. We can create areas of interests (Masks) or use the whole image. In the following example we use global SST images in folder *bsst* to calculate area with SST > 28.5 for all matching image files of 1997 using a mask image *Mask.hdf* having 2 separate masks (areas of interest):

```
wam_count bsst\1997*.hdf 28.5 + mask.hdf Area
```

Output is like that:

```
FileName,AreaLargerThan28.5_1,AreaLargerThan28.5_2
199701.s04m3pfv50-bsst-16b.remap_glob_eq_angle_9km.hdf,0,0
199702.s04m3pfv50-bsst-16b.remap_glob_eq_angle_9km.hdf,25549.35,0
199703.s04m3pfv50-bsst-16b.remap_glob_eq_angle_9km.hdf,170320.1,0
199704.s04m3pfv50-bsst-16b.remap_glob_eq_angle_9km.hdf,1024023,0
…
```

Here is another useful application for *wam_count*. You may have many images that are partly or mostly cloudy and you want to find those that are least cloudy. You can do that and even sort those images so that the best images are at the top of the list. For example, a series of Chl-a images typically have pixels with no data with pixel value 0 or 255. You find the number of

pixels over or below this kind of threshold and sort the images with the *Sort* option. If using the *Mask* option (see below) with the *Sort* option, only the mask value 1 is being used and images are sorted accordingly. For example, assuming that our Chl-a images have the log-Chl scaling of 1-byte per pixel images and pixel value 0 (corresponding to Chl-a of 0.01) means no data. We also assume that we have specified a mask with pixel value 1 in the *Mask.hdf* file. We can then find the number of valid pixels in the masked area of a series of matching image files in the *M2008_chl_day* folder and sort those files (descending order, filenames with the highest number of pixels at the top) with the following command:

```
wam_count  M2008_chl_day\M*mapped.hdf  0.01 +  mask.hdf  sort
```

That would give us a CSV file that starts like that:

```
FileName    CountLargerThan0.01
M2008064_chl_a_mapped.hdf       70427
M2008044_chl_a_mapped.hdf       62055
M2008045_chl_a_mapped.hdf       47052
M2008046_chl_a_mapped.hdf      43866
M2008003_chl_a_mapped.hdf       35014
M2008002_chl_a_mapped.hdf       34711
```

Note that the filenames are sorted according to the number of valid pixels. We now know which our best images are!

wam_trend

```
wam_trend vers. 2.9
Usage1: wam_trend File/Pattern
Usage2: wam_trend File/Pattern [Lin/Sen] [Significance]
  Usage1 is the fast way to calculate linear regression slope image without
    confidence limits of the slope image
  Usage2 is slower but it makes all slopes below threshold confidence limit
       equal to zero
  Either the Linear regression slope or the nonparametric Sen slope is used
  File/Pattern is either a single HDF file with multiple SDS-s of the same size
    or a Pattern of HDF filenames with a SDS of the same size.
  Significance can be 90, 95 or 99 (in %) for Sen test and anything between 50
    and 99% for Linear.
  90, 95 or 99 correspond to 0.1, 0.05 and 0.01 of the two-sided normal
    distribution. Default Significance is 90
Example: wam_trend CHLO\*.hdf Sen 95
```

wam_trend is the updated version of *wam_max_trend*. While the input for *wam_max_trend* was limited to a single HDF file with multiple datasets (output from *wam_annual_max*), *wam_trend* can read input data either from a single file with multiple SDS or a series of matching HDF files. Output file names record the significance level (e.g. 90%, 95% or 99%) that was used. Default is 90% significance. The color scaling is automatic and stretches the colors in between the range of 5% and 95% percentiles of all the estimated slopes of the image. The 5% bottom and the 5% top estimated slope values are capped. The capping is for the case a few very high positive or negative outliers will make the scaling insensitive to the majority of the values.
Examples of usage are in *Detection_of_Change_Arctic_Blooms.pdf*.

wam_variance

wam_variance vers. 1.3

Usage: wam_variance Pattern

where Pattern is a matching pattern of filenames

Calculates pixel-wise mean, standard deviation and the number of valid pixels for a set of matching filenames. The output files (Mean, SD, Valid counts) are saves as HDF.

3.2.2 Convert to HDF/Compress HDF

wam_compress_hdf

wam_compress_hdf vers. 2.3
Usage: wam_compress_hdf PATTERN
where PATTERN is a matching pattern of HDF4 or netCDF filenames
Will overwrite HDF4 files but will create new HDF4 files for netCDF files

wam_compress_hdf reads all matching HDF4 or netCDF files and saves them as HDF4 files with the internal HDF compression. In many cases this results in 10-15 times reduction in file size without a noticeable effect on speed of access. This is a very useful utility in many cases. In addition to reducing the size of a file by many times without practically any penalty, it is valuable in the compositing programs as it removes occasional HDF files without any data to separate folder.

wam_uncompress_hdf

wam_compress_hdf vers. 1.0
Usage: wam uncompress_hdf PATTERN
where PATTERN is a matching pattern of hdf filenames

wam_uncompress_hdf reads all matching HDF files and saves them without the internal HDF compression in a newly created folder *Uncompressed*. This is only needed because some applications cannot read compressed HDF files.

wam_extract_SDS

wam_extract_SDS vers. 2.1
Usage: wam_extract_SDS Pattern SDS [SDS Names or Numbers]
SDSNumber is 0-relative
Examples:
wam_extract_SDS Pathfinder*.nc sea_surface_temperature quality_level
This extracts 2 datasets: 'sea_surface_temperature' and 'quality_level'
wam_extract_SDS Pathfinder*.nc 0 8
This extracts the same 2 datasets numbered 0 and 8

wam_extract_SDS extracts specified individual datasets (SDS) from all matching HDF and netCDF (.nc) files and saves them in as HDF4 files. You can specify the datasets to be extract by

either their names or by the sequence numbers (0-relative). If any of the names has space then you cannot use the names and must use sequence numbers. Use space to separate the numbers or names of the requuired datasets, e.g. 0 2. The numbering is 0-relative. The examples extract the same two datasets (*sea_surface_temperature* and *quality_level*) specified by either by names or sequence numbers.

wam_assemble_hdf

```
wam_assemble_hdf vers. 1.1
Usage: wam assemble_hdf PATTERN
      where PATTERN is a matching pattern of hdf filenames
```

wam_assemble_hdf reads all matching HDF files, extracts the first dataset (SDS) and saves these in a new HDF file with multiple datasets. For example, if you have multiple monthly (or daily) HDF files and you want to assemble those into a single yearly (or monthly) HDF file then this utility does just that. For example, the following command reads 12 matching monthly HDF files and makes a single annual HDF filed called *S1998_assembled.hdf*:

```
wam_assemble_hdf   Monthly\S1998*.hdf
```

wam_disassemble

```
wam_disassemble vers. 2.2
Usage: wam disassemble PATTERN
  where PATTERN is a matching pattern of HDF files with multiple SDSs
```

wam_disassemble is the opposite for *wam_assemble_hdf* . It reads matching HDF files with multiple datasets (SDS) and breaks each into a set of multiple HDF files with a single dataset in it. For example, *wam_annual_max* finds the annual maximum for each year and saves in a single HDF file (*Max.hdf*) with a dataset for each year. In order to make a time series with *wam_statist* that assumes each dataset in a separate file, we need to break up the *Max.hdf* file into separate files for each year and we can do that with *wam_disassemble*. A similar task is to break up the monthly Means file (output of *wam_anomaly*) into 12 separate monthly files.

wam_hdf2nc

```
 wam_hdf2nc vers. 1.0
 Usage: wam_hdf2nc FileNamePattern [Mirror]
   where FileNamePattern is a matching pattern of HDF4 filenames
 Optional argument Mirror makes mirror over horizontal axis
```

wam_ hdf2nc reads all matching HDF4 files, extracts all datasets (SDS) and saves these in a new netCDF file with the *.nc* extension. Example:

```
wam_ hdf2nc  Monthly\V2014*.hdf
```

wam_nc2hdf

```
wam_nc2hdf vers. 1.2
Usage: wam_nc2hdf   FileNamePattern
  where FileNamePattern is a matching pattern of netCDF filenames
```

wam_ nc2hdf reads all matching netCDF files, extracts all datasets (SDS) and saves these in a new HDF4 file with the *.hdf* filename extension. Example:
wam_ nc2hdf Monthly\V2014*.nc

wam_convert_100xbyte

wam_convert_100xbyte vers. 1.1
Usage: wam_convert_100xbyte Pattern

Reads all matching HDF files, converts Float32 datasets (SDS) of relative concentration (range 0.0-1.0, e.g. with ice concentration) to scaled byte (slope = 0.01), adds Start and End attributes based on file name.

wam_convert_amsre

Usage: wam_convert_amsre FileNamePattern [n]
where FileNamePattern is a matching pattern of AMSR-E filenames
n (<= 5) is the sequence number of an image to be read:
SST = 0, WSPD = 1, VAPOR = 2, CLOUD = 3, RAIN = 4
The default is to read all 5 images into a single HDF file.

wam_convert_ascat

wam_convert_ascat vers. 1.0
Usage: wam_convert_ascat FileNamePattern
where FileNamePattern is a matching pattern of filenames
Converts ASCAT netCDF files to HDF, replacing Latitude and Longitude in Int32
format with Float32 format
Example:
wam_convert_ascat C:\Sat\ASCAT\ascat_2010*.nc

Reads all matching ASCAT netCDF files, converts Latitude and Longitude in the unconventional Int32 format to conventional Float32 format, saves all in HDF files using SDS (Scientific Datasets). ASCAT is a scatterometer that provides Level-2 ocean wind products; data are available at http://podaac.jpl.nasa.gov/DATA_CATALOG/ascatinfo.html), Gridded and composited (Level-3) wind data are easier to use and are available at ftp://podaac.jpl.nasa.gov/ocean_wind/ccmp/L3.5a/data/; see wam_convert_ccmp.

wam_convert_avisosla

wam_convert_avisosla vers. 1.1
Usage: wam_convert_avisosla FileNamePattern
where FileNamePattern is a matching pattern of netCDF filenames

Reads matching AVISO sea-surface height (sea-level anomaly) files and converts to HDF4. AVISO data can be downloaded from ftp://ftp.aviso.oceanobs.com.

wam_convert_ccmp

Usage: wam_convert_ccmp_wind Pattern
- converts CCMP Wind data in netCDF to HDF .

Reads all matching Cross-Calibrated Multi-Platform (CCMP) Ocean Surface Wind Components (Atlas et al. 2008, 2009) and converts the Level 3.5a net CDF files into HDF. CCMP data are multi-satellite products of ocean surface wind that span nearly 21 years. The product is derived through cross-calibration and assimilation of ocean surface wind data from SSM/I, TMI, AMSR-E, SeaWinds on QuikSCAT, and SeaWinds on ADEOS-II. CCMP data can be downloaded from ftp://podaac.jpl.nasa.gov/ocean_wind/ccmp/L3.5a/data/.

wam_convert_cm

wam_convert_cm vers. 1.2
Usage: wam_convert_cm Pattern

Reads all matching Surface incoming shortwave radiation datasets from the EUMETSAT Satellite Application Facility (SAF) network Climate Monitoring (CM SAF, http://www.cmsaf.eu) of the Metosat disk area (1983-2013) in netCDF format, adds projection and converts to HDF4.

wam_convert_goes1112
wam_convert_goes1112 vers. 1.0
Usage: wam_convert_goes1112 Pattern

Reads all matching GOES 11-12 raster binary files with the PODAAC header, converts to HDF with 2 datasets: SST and the scaled cloud probability. Adds attributes for time series analysis. The cloud percentage is in a scaled byte with the following scaling: P(clear) = 1.002 - exp[(2 - count) / const] where const = 40.546121. For example, pixel value 252 corresponds to cloud probability of 0.01%, i.e. most likely a clear pixel, pixel value 237 corresponds to cloud probability of 0.1%. See ftp://podaac.jpl.nasa.gov/sea_surface_temperature/goes/ for GOES 11-12 data and documentation. This application works with the new format of GOES 11-12 data using daily datasets with filenames like sst24b_2008_001, sst24b_2008_002, etc. The string "24b" in the file name means 24 hr SST with the Bayesian cloud probability. "2008" means year and "001", "002" mean year days.

Another WAM application, *wam_screen_goes1112* screens the output of *wam_convert_goes1112* and keeps only those pixels above a threshold.

wam_convert_log
Usage: wam_convert_log Pattern [slope [intercept]]
Defaults: slope=0.018, intercept=0; These are for PP
For Log-Chl use slope=0.015, intercept=-2

wam_convert_dim
Usage: wam_convert_dim Pattern [param1 [param2]]

Reads matching MERIS DIM files, converts the specified datasets to SDS (scientific datasets) in HDF4 format; adds longitude and latitude arrays for geo-location.

wam_convert_k2c
wam_convert_k2c vers. 1.0

```
Usage: wam_convert_k2c FileNamePattern [SDSnumber]
  where FileNamePattern is a matching pattern of HDF/netCDF
   filenames
Optional argument SDSnumber reads the 0-referenced SDS
Default SDSnumber is 0
```

Reads matching HDF or netCDF files, selects a single dataset (SDS) and converts values in Kelvin to Celsius in the SST-Pathfinder scaling. The default is to read SDS=0 but that can be changed with optional attribute SDSnumber.

wam_convert_mld

```
Usage: wam_convert_mld Pattern [landmask.hdf]
```

Reads all matching HDF files, converts Float32 datasets (SDS) of to Int16.

wam_convert_n1

```
wam_convert_n1 vers. 1.3
Usage1: wam_convert_n1 FilePattern [Variable1 [Variable2] ...]
- Extracts selected datasets from matching MERIS *.N1 files and saves in HDF4
 with added attributes
Usage2: wam_convert_n1 FilePattern
- Shows a list of datasets in the 1st matching file
NOTE!!! If you get 'Couldn't read all the bytes - Cancel and release buffer?'
- you have to open WIM, select 'Settings-Misc' and set Image Header length to 0
bytes.
```

Reads matching MERIS (ENVISAT) N1 files, converts selected variables to SDS (scientific datasets) in HDF4 format; adds longitude and latitude arrays for geo-location. Note: if you get a message “Couldn’t read all the bytes – Cancel and release buffer?” then you have to open WIM, select *Settings – Misc* and set the *Image Header* length to 0 bytes.

wam_convert_ncsst

```
wam_convert_ncsst vers. 1.0
Usage: wam_convert_ncsst Pattern
 where Pattern is a matching pattern of filenames;
 only the 1st dataset = SST is read
```

Converts global 6-km SST data from GHRSST netCDF files into HDF4 files. A major problem with high-resolution infrared satellite data is that in cloudy conditions there are no data. A project called “OSTIA Sea Surface Temperature and Sea Ice Analysis” tries to remedy this problem by combining low-resolution data from microwave sensors available even through clouds with higher-resolution data from infrared sensors. For example, some of the merged SST data has been derived from the following sensors: AMSRE, ATS_NR_2P, AVHRR18_G, AVHRR17_NAR, AVHRR18_NAR, OSISAF_ICE, SEVIRI and TMI. The data are merged using sophisticated algorithms. The 6-km global data are provided by the UK Met Office as netCDF files. Each file contains multiple datasets (SDS): *analysed_sst*, *analysis_error*, *sea_ice_fraction* and *mask*. Only the 1st dataset (*analysed_sst*) is extracted and saved as HDF4. The original daily data files can be downloaded from

ftp://podaac-ftp.jpl.nasa.gov/allData/ghrsst/data/L4/GLOB/UKMO/OSTIA/. Select the bz2-compressed daily *.nc files (e.g. *20120808-UKMO-L4HRfnd-GLOB-v01-fv02-OSTIA.nc.bz2*), download and uncompress before running *wam_convert_ncsst*. Uncompression can be done either in command line (*bzip2 –d *.bz2*) or using a GUI program like *7z*.

wam_convert_ngsst

Usage: wam_convert_ngsst Pattern

Converts the New Generation SST (NGSST) binary datasets to HDF while adding attributes. NGSST dataset is being merged from various sources at Tohoku University (Dr. Kawamura) and is available around Japan (see paragraph 18 in *WIM.pdf*).

wam_convert_oisst

Usage: wam_convert_oisst FileNamePattern [0,...]
where FileNamePattern is a matching pattern of filenames
If no additional arguments are given then all 4 datasets are read:
0 = SST, 1 = SST anomaly with respect to 1971-2000,
2 = standard deviation of the analysis error, 3 = sea ice concentration
You can specify the datasets to be read, e.g. 0 for SST, 1 for SST anomaly...
For example,
wam_convert_oisst amsr*.* 0 1
will read only the SST and SST anomaly datasets and save in a single HDF file.

Converts the Optimum Interpolation (OI) SST (OISST) binary datasets to HDF4 while adding attributes. See Reynolds et. al 2006, Daily High-resolution Blended Analyses. (ftp://eclipse.ncdc.noaa.gov/pub/OI-daily-v2/daily-sst.pdf) for a description of the procedure. Daily IOSST data are available at: ftp://eclipse.ncdc.noaa.gov/pub/OI-daily-v2/IEEE/. The advantage of the OI datasets is that they have no missing pixels due to clouds that are a major problem in cloudy regions when using infrared SST sensors.

wam_convert_percent

wam_convert_percent vers. 1.3
Usage: wam_convert_percent Pattern

Reads all matching HDF files, converts percent (range 0.0-100.0%) to fraction in a scaled byte (e.g. 0.0-1.0).

wam_convert_ssmi

wam_convert_ssmi vers. 1.0
Usage: wam_convert_ssmi Pattern

Reads all matching raster binary files of ice concentration distributed by NSIDC (ftp://sidads.colorado.edu/pub/DATASETS/seaice/polar-stereo/nasateam/) and converts to HDF files with added attributes, scaling and projection. Ice concentration is represented as fraction of ice with a valid range of 0.0-1.0.

wam_convert_to_lin

wam_convert_to_lin vers. 1.3

Usage: wam_convert_to_lin Pattern [slope [intercept]]
Defaults: slope=0.5, intercept=0
 Defaults are suitable for PAR.

Reads all matching HDF files with float32 datasets and converts the first dataset to byte with linear scaling. The default slope (0.5) and intercept (0) of the linear scaling are suitable for converting float32 datasets of PAR to byte. This saves a lot of disk space without losing significant accuracy of the data.

wam_convert_to_power

wam_convert_to_power vers. 1.0
Usage: wam_convert_to_power Pattern Power
Pattern is a filename pattern to match
Power is the power to be used, e.g. 2 for square, 3 for cube, etc

Reads all matching HDF files and converts the first dataset to a power of, e.g. 2 or 3. Implemented specifically for converting wind speed to wind stress.

wam_gradient

wam_gradient vers. 1.1
Usage: wam_gradient Pattern [Vert or Hor]
 Pattern is a filename pattern to match
 default is Vert = vertical or north-south, Hor = horizontal or east-west

Calculates vertical (NS) or horizontal (EW) gradients of matching HDF files. This is intended to calculate wind stress divergence (meridional or zonal) when using CCMP wind stress data, e.g. UPSTR for zonal divergence or VPSTR for meridional divergence (see *wam_convert_ccmp*).

wam_mirror

wam_mirror vers. 1.1
Usage: wam_mirror Pattern h/v/b [Projection [ProjectionShift]]
 Mirror options: h = horizontal, v = vertical, b = both
 Optionally set Projection and ProjectionShift:
 currently only Glob Equal Angle = GEA is implemented.
Examples:
 wam_mirror *.hdf h
 wam_mirror *.hdf h GEA 180

Reads all matching HDF or netCDF (*.nc) files, mirrors over horizontal, vertical or both axes. Optionally sets projection type (currently only to Global Equal Angle) and projections shift (for Global Equal Angle), saves as HDF.

wam_replace_pixel_values

wam_replace_pixel_values vers. 1.1
Usage: wam_replace_pixel_values file_pattern [pixelValueFrom] [pixelValueTo]
Default values are: pixelValueFrom=255, pixelValueTo=0

Reads all matching HDF or netCDF (*.nc) files, replaces unscaled pixels with a new unscaled pixel value. The default option changes all pixel values 255 to 0. This makes white pixels black in a typical Log-Chl color scaling.

wam_replace_values

```
wam_replace_values vers. 1.1
Usage: wam_replace_values Pattern FromRange ToRange NewValue
```

Reads all matching HDF or netCDF (*.nc) files and replaces pixels with scaled (geophysical) range values from *FromRange* to *ToRange* with *NewValue*. This is similar to the *Transf-Replace Values* function in WIM. Note that the related command *wam_replace_pixel* values works with unscaled pixel values.

wam_lin_transform

```
wam_lin_transform vers. 1.0
Usage: wam_lin_transform Pattern A [B] [float]
  Pattern is a filename pattern of HDF files to match
  A and B are coefficients according to Y = A * X + B
  If B is not given, it is assumed to be 0.
  'float' is an option that directs to save the output in Float32 format
  The default is to keep the original format (Byte, Int16, Uint16, Float32) and scaling.
  Use the 'float' option if the result Y becomes too big to store in the original format and scaling.
Examples:
  wam_lin_transform C:\Sat\S*.hdf 1.5 0.5
  wam_lin_transform C:\Sat\S*.hdf 100 5 float
```

Reads all matching HDF or netCDF (*.nc) files, replaces valid pixels with their linear transform according to equation Y = A *X + B, where X is the old value and Y is the new value. The default is to keep the original image format (e.g. Byte, Uint16, Int16, etc) with the original scaling. If the new pixel value becomes too big to be scaled using the original format and scaling then you can convert the output into Float32 without any scaling. This command is similar to the *Transf-Linear Trans* function in WIM.

3.2.3 Operations with bands

wam_band_ratio

```
wam_band_ratio vers. 4.17
Usage: wam_band_ratio Rrs555_pattern AlgorithmType
Rrs555_pattern is name pattern for the denominator band, e.g.
  Rrs555, Rrs565, Rrs551, or Rrs547
It is assumed that all required bands are in the same directory
AlgorithmType can be: CHLCAL2015, CHLCALFIT3, CHLCALFIT4, OC2, OC3, OC4, OC3L, O
C4L,
   CHLSPGANT3, CHLSPGANT4, SPGANT3BLENDED, SPGANT4BLENDED, ZEUSPGANT
```

CHLCALFIT3 and CHLCALFIT4 are OC3/OC4 type fits to California Current Chl;
CHLCAL2015 are updated MODISA and VIIRS OC3 type fits;
CHLSPGANT3 and CHLSPGANT4 are Southern Ocean Chl-a algorithms.
CHLSPGANT3BLENDED and CHLSPGANT4BLENDED blend with standard algorithm and
save also MaxBandRatio file that can be used for blending other products.

wam_band_ratio implements band-ratio algorithms for Chl-a or CDOM. OC4/OC3 are the standard chlorophyll algorithms (O'Reilly et al., 1998) that can be downloaded directly from NASA but *wam_band_ratio* allows creating products with other algorithms. OC4L is the Arctic version (Cota et al., 2004) and SPGANT is a Southern Ocean specific Chl-a algorithm. The different remote sensing reflectance (*Rrs*) bands (e.g. 443, 490, 510, 555 nm) must be in the same or separate folders. The output, Chl-a images scaled in byte or CDOM images scaled in Int16 are saved in HDF4 format.
It must be noted that ideally the band-ratio and other bio-optical algorithms are applied to Level-2 radiances, i.e. before binning, compositing and mapping. However, it may be too much work to obtain individual Level-2 images and apply the various algorithms.

wam_ratio_2sets

wam_ratio_2sets vers. 1.0
Usage: wam_ratio_2sets Pattern1 Pattern2
Pattern1 and Patteren2 are filename patterns to match
The result is a pixelwise ratio of File1/File2

wam_ratio_2sets uses 2 sets of matching HDF files and calculates a set of ratio images. For example, if a *Pattern1* of *S2009*.hdf* matches with 12 images and a *Pattern2* of *S2008*.hdf* matches with 12 images then the result would be 12 ratio images of the 2009 data over 2008 data. Each of the ratio images is a *Float32* pixel image and only those pixels that are valid in both image 1 and image 2 have the ratio value calculated while the pixels corresponding to invalid pixels are set to 0.

wam_blend_spgant

wam_blend_spgant vers. 3.3
Usage: wam_blend_spgant SpgantPattern StandardPattern
SpgantPattern is name pattern for the SPGANT files
StandardPattern is name pattern for the standard (e.g. OC4 or CALFIT) files
It is assumed that the MBR files are in the same folder with SPGANT
The front position is specified in a file 'stf_-180_180_no_loop_1deg.csv' that has to be in the
Wimsoft, Wimsoft\Maps or Wimsoft\VOs folder

wam_blend_spgant blends 2 sets of images using another image of the maximum band ratio (MBR). It follows the method of Kahru, M. and B.G. Mitchell (2010), Blending of ocean colour algorithms applied to the Southern Ocean, *Remote Sensing Letters*, 1: 2, 119-124, doi: 10.1080/01431160903547940. PDF. The front position is taken from a file 'stf_-180_180_no_loop_1deg.csv' that has to be either in the *Wimsoft*, *Wimsoft\Maps* or *Wimsoft\VOs* folder. It specifies the mean position of the Subtropical front according to Orsi et al. (1995). The first set of images are using the Southern Ocean specific algorithm (SPGANT) whereas the 2nd set of images are standard (e.g. OC4 or CALFIT) images. The same procedure can be applied to Chl-

a, NPP and other variables. The procedure can be adapted for other cases that need blending of two or more algorithms.

3.2.4 Compositing

wam_composite

```
wam_composite vers. 2.3
Usage: wam_composite List_or_Pattern Composite [Count
[SDS_Number]]
```

wam_composite is a quick way to create image composites from various file types. Detecting the range of valid values and excluding flagged values is important for creating proper composites. You may need to check that the proper values for *fMin* and *fMax* are detected. Here is a typical example where to use *wam_composite*. Let's say that you want to make a composite Chl image of a certain 10-day period using available daily images but the official products are daily and 8-day images. You can download the daily images and make the required composite yourself. You can make the composites interactively with WIM but it is much more convenient to use *wam_composite*. All you need to do is make a list of the image files names that you want to composite. Having a list gives the additional benefit of having a record of files used in the composite. You can create the list by simply doing dir/b on your image files and piping that to a list file. For example,

dir/b S200301*DAY_CHL* > list.txt

That creates a list file called *list.txt* and dumps all the matching SeaWiFS filenames into it. Please note that if you want to use the list file from another directory then it should have the full paths and not just the filenames. You can use the option /s to add the full path to the list. Let's say that we want to create a composite image of ALL the January images of different years. We can create a list of all SeaWiFS January images with a command like this:

dir/b /s S????001*DAY_CHL* > list.txt

That dumps all the matching filenames into *list.txt*. You can then remove some of the files from being used in the compositing by preceding the file name with the # character. That allows you to keep the file name in the list in case you want to use it another time.

As output, two HDF files are created: a *composite* file (representing the mean of valid pixels) and the *count* file (showing the number of pixels that were used for each composited pixel). The determination if the pixel is valid or not depends on the file format and multiple conventions. If the valid range is not determined correctly then the composited image is not correct. The valid range is printed for the first image to be used. You should check the output and confirm that the valid range is correct.

If you want to use the composited image in a time series analysis then you need to make sure that the attributes are correct. In the resulting images the attributes are set so that the start attributes (e.g. "Period Start Year", "Period Start Day", "Start Time", "Start Year", "Start Day", etc.) are taken from the first image and the end attributes from the last image. The attributes "Start Year", "Start Day" and corresponding end attributes are used in time series analysis. Different satellite images have slightly different ways of recording the time and it is possible that the attributes are not read correctly.

Many HDF files contain multiple datasets and you probably want to read only a specific dataset and composite those from a series of HDF files. Therefore there is an option to specify the SDS (Scientific Dataset) number that you want to read and composite. The default SDS number is 0.

wam_composite_pairs

```
wam_composite_pairs vers. 1.1
Usage: wam_composite ListOf2Files
   ListOf2Files is a CSV text file with a separator of comma, tab
   or space
```

wam_composite_pairs makes composites of each pair of filenames read from a comma separted list. For example, we may have separate images of ice concentration in the Northern and Southern hemisphares mapped to the same global projection. We then want to combine those 2 into a common global image. For each pair a new composite is made. The list must have full pathnames for both files. A sample list file looks like this:

C:\Sat\North\img1A.hdf, C:\Sat\South\img1B.hdf

C:\Sat\North\img2A.hdf, C:\Sat\South\img2B.hdf

C:\Sat\North\img3A.hdf, C:\Sat\South\img3B.hdf

C:\Sat\North\img4A.hdf, C:\Sat\South\img4B.hdf

This will produce 4 composited files; one for each pair using the first image in each file. The filenames of the new composites are combined from the names of the two input files.

wam_composite_quikscat

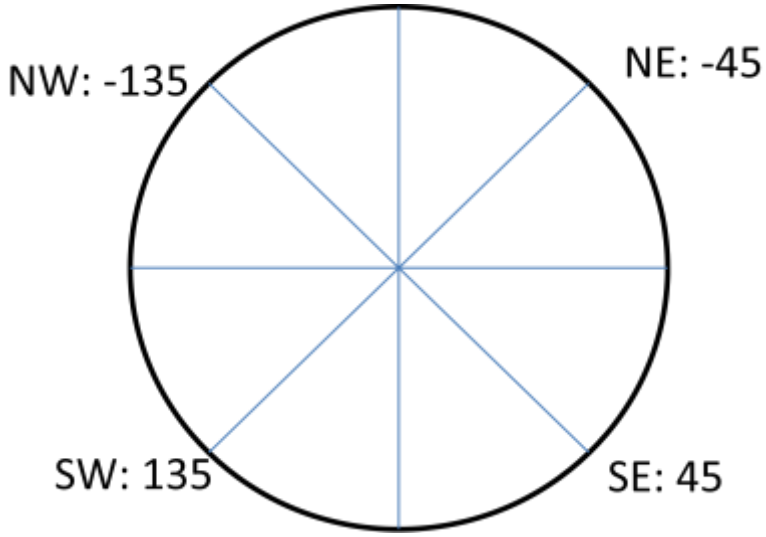
```
wam_composite_quikscat vers. 2.0
Usage: wam_composite_quikscat Pattern [u/v]
   where Pattern is matching filename pattern
   optional u/v specifies either wind u or v component
   the default (without u or v) is wind speed
```

wam_composite_quikscat composites the ascending and descending orbits of Level-3 QuikSCAT files and saves as HDF. The default option is to composite wind speed but either the *u* or *v* component can be selected.

wam_rotate

```
wam_rotate  vers. 1.0
Usage: wam_rotate Pattern1 Pattern2 degrees
where degrees is the rotation angle in degrees
If images are of different size then Pattern2 will be remapped to
   Pattern1
Optional arguments:
SaveBoth=Yes saves both components, i.e. rotated by Degrees and
   its orthogonal
```

wam_rotate reads pairs of datasets corresponding respectively to the eastward and northward wind components and makes a 2D rotation to calculate the wind components in other directions. Rotation of a vector is considered positive in counterclockwise but as we are rotating the axis in the opposite direction. For example, calculating the wind component in the NW-direction uses -45 degrees (negative 45), in the SE-direction 45 degrees, in the NW-direction -135 degrees (negative 135) (see Fig. below). Optionally the orthogonal component is also calculated.



3.2.5 Edge or Front Analysis

wam_edge

```
wam_edge vers. 1.9 by mkahru@ucsd.edu
Usage: wam_edge Pattern [WindowSize [LandMask [Left Top Right
Bottom]]]
  Use windowSize 0 for variable window size
  LandMask is an HDF image with pixel values 0 for ocean (to be
used for edge
    detection) and all other pixels to be excluded.
  Left Top Right Bottom are pixel coordinates and allow to cut
out a
  subset of the image (must equal in size to LandMask!)
```

wam_edge finds edges (fronts) in a series of SST images. Please see a separate document *Exercises_WAM_Edge_Detection.pdf* for detailed instructions and examples.

wam_edge_accumulate

```
wam_edge_accumulate vers. 2.2
Usage: wam_edge_accumulate Pattern 1/12 MinValid
  where Pattern is the pattern of source files and NOT the
*Sied*.hdf files
  Both the source AND the Sied files must be in the same folder!
  Source files are needed to get the valid pixels in order to
calculate the frequency of fronts (FF) per valid pixels.
  12 or 1 means that averaging is either per Month (12) or for
Overall (1)
  Optional MinValid specifies the minimum valid count required to
calculate FF.
  Default is MinValid=1, i.e. at least 1 valid count is needed to
calculate FF.
```

```
  This may give abnormally high FF if valid count is very small.
  Setting MinValid to a higher value, e.g. 3 or 5 reduced the
chance of spurious high FF values.
```

Please see a separate document *Exercises_WAM_Edge_Detection.pdf* for detailed instructions and examples.

3.2.6 Image Operations

wam_contours

wam_contours vers. 1.0
Usage: wam_contours Pattern From To Step PixValue

wam_contours automates the operation *Examine-Contour* Lines in WIM. It can be applied to a set of HDF image and needs the following input: *From* (starting isoline value), *To* (ending isoline value), *Step* (isoline step size), *PixValue* (value of the isoline to be drawn). Example:
wam_contours A*.hdf 0.5 0.5 1.0 0.01
The example above tries to make a single isoline (at 0.5) with pixel value 0.01 (black if using *Log-Chl* scaling). The *Step* value is irrelevant in this case as both *From* and *To* have the same value (0.5).

wam_mozaic_land_ocean

```
wam_mozaic_land_ocean vers. 1.0
Usage: wam_mozaic_land_ocean OceanImage LandImage Landmask
```

wam_mozaic_land_ocean is a utility to make a mosaic from RGB images of land and ocean based on a land mask image. When making quasi-true color images (e.g. from MODIS 250 m data) that is normally very bright compared to the dark ocean. In order to see features in the ocean we need to enhance the brightness but then land becomes too bright and loses its structure. The solution is to use separate brightness (color stretch) intervals for land and ocean and then make mosaic of the two images. We use a land mask image that defines where to use the pixel from the land image (land mask pixel value > 0) or ocean image (land mask pixel value = 0). All three images must be of the same size.

wam_overlay

wam_overlay vers. 3.7
Usage: wam_overlay Pattern OverlayPattern
Pattern is a pattern of base files
OverlayPattern is either a single file or a Pattern of overlays.
If the OverlayFile is RGB image then all images will be converted to RGB
Options:
Lut=file.lut sets external LUT file
ColorMin=min and ColorMax=max set pixel values of color stretching
If colorMin is given but no colorMax, then colorMin is interpreted
as the Max in Color Scaling for multiByte datasets.
xPos=X sets x position of annotation, yPos=Y is y position of annotation

wam_overlay is a utility to automate the creation of overlaid files. For example, you can create a standard overlay image with coastlines, land masks, grid, color scale, station locations, etc and

put it on top of a series of images. If the HDF file with images has multiple datasets (SDSs) then each one of those will be saved separately with a numerical index. In another case, you may have a series of base images (e.g. monthly chlorophyll composites) and you want to overlay each one with the respective image of ice edge (or SST fronts, or anything else that is different from one image to another). In that case you have 2 sets of images: those of the base images and of the overlay images. You can make a batch file where you specify each overlay file separately for each base image file but that would be a lot of work if you have many images. *wam_overlay* automates the matching process as it will read the times of each base image and will match with the nearest in time image in the set of overlays.

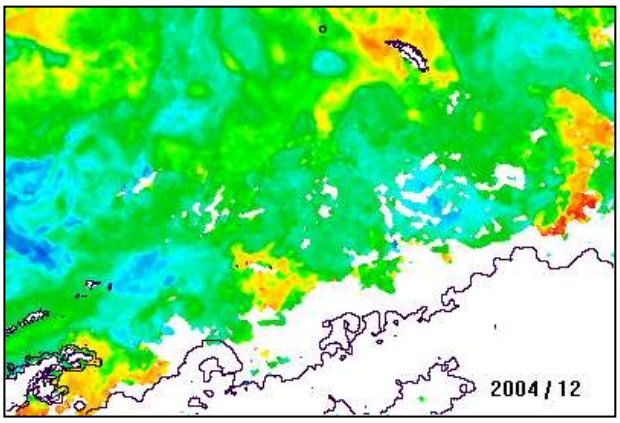
Pattern is a matching pattern of file names to use as the base images. *OverlayPattern* specifies either a single overlay file or a pattern of matching overlay files (images) which will be matched with the base images according to their timing. For example, you may have a January, 2011 monthly chlorophyll image will be matched with the January, 2011 ice edge image. *LUT* specifies a LUT file, *colorMin* and *colorMax* can be specified to pick color range for color stretching. Only *Pattern* and *OverlayFile* are required, other arguments are optional. Annotation (Date) will be added id its position is specified by xPos=X and yPos=Y (where X and Y are pixels from left or top, respectively). All input and output image files are assumed to be HDF.

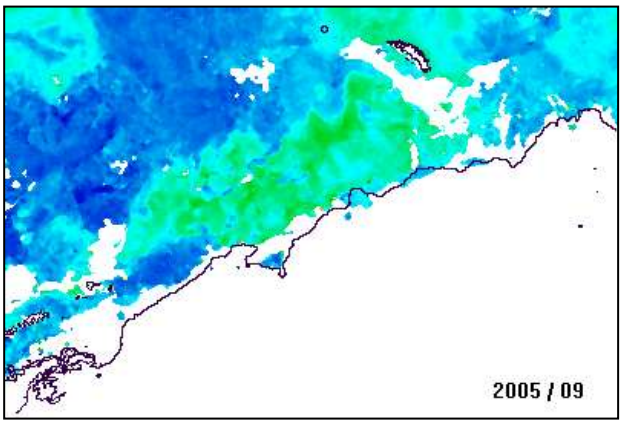
For example, command below overlays a single overlay image (in *Overlay.hdf*) to set of matching files (*A*.hdf*) and adds an annotation at specified location:

```
wam_overlay A*.hdf  Overlay.hdf xPos=318 ypos=250
```

The command below overlays each base image (*A*.hdf*) with a respective overlay image (picked from a set of *Overlay_*.hdf*) and adds an annotation at specified location. Two examples of monthly Chl images overlaid with respective ice edge images are shown below.

```
wam_overlay A*.hdf  Overlay_*.hdf xPos=318 ypos=250
```





wam_reduce

```
wam_reduce vers. 2.4
Usage: wam_reduce Pattern [N] [RedyceType]
  Pattern is a matching pattern of HDF file names
  N is times of reduction, default is 2
  Default ReduceType is Median
Possible ReduceTypes: Average, Min, Max, Median, Neighbor, Sum
```

wam_reduce is a simple example how to use WAM to read a series of images, reduce the size of the images and save the reduced images. The most common usage is converting between global standard mapped images (SMI) of different sizes. For example,

wam_reduce C:\temp\A*.hdf 4 Max

reduces all matching *A*.hdf* images in source images in *C:\temp* by 4 times using the maximum pixel value in the 4 x 4 pixel window. As output, the program generates reduced HDF files in the current directory. The example can be easily modified or used as a building block in other applications. For more advanced options please use *wam_reduce_valid*.

wam_reduce_valid

```
wam_reduce_valid vers. 2.3
Usage: wam_reduce_valid Pattern
Pattern is a matching pattern of HDF4 or netCDF file names
Options:
ReduceX=X sets times of reduction, default is 2
ReduceType=type sets ReduceType. Default is Average
Possible ReduceTypes: Average, Min, Max, Median, Neighbor, Sum
sds=X specifies to read the Xth dataset instead of the default 0th
sdsName=N specifies the name of the SDS to read instead of the default 0th
MaskImage=name is HDF4 file that specifies valid area with pixel value > 0,
  area masked with 0 is filled with FillValue
scaling=logChl converts output to byte with LogChl scaling
```

wam_reduce_fixed

```
wam_reduce_fixed vers. 1.0 by mkahru@ucsd.edu
Usage: wam_reduce_fixed Pattern [RedyceType]
   The resulting image size is 192 x 94
   Default ReduceType is Max
Possible ReduceTypes: Average, Min, Max, Median, Neighbor, Sum
```

wam_reduce_fixed is a another version of the *wam_reduce* utility. Instead of reducing a certain number of times, it reduces to a fixed image size. The final image size can only be changed in the source code. When doing the reduction, it uses also uses the valid range as *wam_reduce_valid*.

3.2.7 Mapping

wam_remap

```
wam_remap  vers. 2.7
Usage: wam_remap Pattern TargetProjectionFile [SDS_numbers] [JPG][PNG]
  SDS numbers are separated by space. Default is to Remap only SDS 0.
Note that SDS 0 can be Latitude that you probably DO NOT want to remap.
Example: wam_remap A*.hdf ..\Target.hdf 0 1 5
  will remap SDSs 0, 1, 5 in all matching files to the projection in Target.hdf
 and save as HDF
Options JPG or PNG save additionally the JPG or PNG of only the 1st SDS.
```

wam_remap2

```
wam_remap2 vers. 2.4
Usage: wam_remap2 Pattern Target [SDSNumber MedFilt FillSize FillValue Palette]
Default is to read SDS number 0, do no filtering, no filling and
  using the default palette
Optionally can read a specified SDS (0-referenced), Median-filter, Fill-holes
  and use a selected palette file
SDSNumber is the number of SDS (0-referenced) to be read and remapped
MedFilt is the median window size, typically 3 (skipped if 0 or negative)
FillSize is the Fill Holes window size, typically 1 or 3 (skipped if 0
  or negative)
Fillvalue is the Fill Holes pixel value, e.g. 0 or 255 (skipped if negative)
Palette is the LUT file
Option: convert=pv  converts to Byte pixel value
From=X To=Y color stretch from X to Y of byte values
```

wam_remap and *wam_remap2* are two related utilities for mapping HDF files. Simple and easy remapping is also included in the GUI utility *wam_series*. Both *wam_remap* and *wam_remap2* can remap HDF files which have geo-referencing information either in the file itself or in an external geo-referencing file. The name of the external geo-referencing file is guessed from the filename itself. Another sample program, *wam_remap_lla* uses a specified list of files and their respective geo-referencing files for remapping. The main difference is that with *wam_remap* you can remap **multiple** datasets (SDS) from the same file while with *wam_remap2* you have more options but remap only one dataset from a file at a time.

Pattern is the filename or filename pattern (with wildcard characters ? and *) that matches one or more files. *TargetProjectionFile* is the filename of a HDF file in the target projection. The optional *SDS_numbers* are a 0-referenced SDS (dataset) sequence number to be mapped. The default (if none is given) is SDS number 0.

For example, you can try to remap a sample SeaWiFS Level-2 image to the *Linear* projection in the example file composite.hdf:

wam_remap S2003090*.sub.hdf ..\Examples\composite.hdf 2

The above command assumes that you have the SeaWiFS Level2 file in the current folder and the target projection file in a folder *Examples* one level up. Please note that we have specified SDS (dataset) number 2, i.e. *chlor_a*. If you open the file with WIM you can realize that datasets 0 and 1 are, respectively, *longitude* and *latitude* and you cannot remap those. Please note that omitting the SDS number means that you pick SDS number 0. In this case that means *longitude* and an empty resulting image. You can remap multiple datasets at once by specifying their sequence numbers (0-relative) separated with a space. For example

wam_remap A*.hdf ..\Target.hdf 2 4 6

will find all matching *A*.hdf* files, and remap datasets 2, 4 and 6 to the projection taken from file *..\Target.hdf*.

The syntax of *wam_remap2* has more options:

```
Usage: wam_remap2 Pattern Target [SDSNumber MedFilt FillSize
FillValue Palette]
```

For example

wam_remap2 MYD02QKM.*RGB*.hdf ..\sb_aco_200m.hdf

will find all matching MODIS-Aqua RGB files, read the first dataset (number 0) and remap it to the projection taken from file ..\sb_aco_200m.hdf. The program will try to find the respective external geo-referencing file assuming MODIS filename nomenclature. The external geo-referencing file is also called LLA (Longitude-Latitude Array) file and is assumed to have name pattern MO03* or MY03*. If there are multiple matching LLA files then the first one is used. You can create your own LLA file with a WAM program *zoom_modis_lat_lon* (see below). If present, the output from *zoom_modis_lat_lon* is used as it has higher resolution (250 m). *Wam_remap2* will pick the LLA file generated by *zoom_modis_lat_lon* simply because it is listed (in alphabetic sequence) before the standard MODIS product 03. The remapped output file will have the name of the target projection file in it. Therefore the same image mapped to different projections will be in separate files.

The optional arguments *MedFilt* specifies the window size (typically 3) of the median filter (skipped if negative), *FillSize* specifies the window size (typically 1) of the fill holes operation (skipped if negative), *FillValue* specifies the pixel value to be filled with neighboring pixels (typically either 0 or 255) and *Palette* is the optional external palette file **.lut* (as found in the LUT folder).

wam_remap_all

wam_remap_all vers. 1.0
Usage: wam_remap_all Pattern TargetProjectionFile
Example: wam_remap_all Data\A*.hdf Target.hdf

wam_remap_all remaps all datasets in a series of matching HDF4 or netCDF files to a target projection given in a HDF4 file.
For example, the following command remaps all datasets in matching A*.hdf files to the projection in myTarget.hdf:

wam_remap_all A*.hdf ..\myTarget.hdf

wam_remap_lla

wam_remap_lla vers. 2.2
Usage:
wam_remap_lla list_file target_file
where list_file has 2 filenames separated by comma, tab or space per line
Example:
wam_remap_lla list2.txt mytarget.hdf
If the same LLA is used for all files, you can use:
wam_remap_lla Pattern LLA_file target_file
where Pattern is a matching filename pattern,
LLA_file is a HDF file with the Latitude and Longitude arrays,
output is 1 level up from the source file location
Example:
wam_remap_lla *.hdf ..\myLLA.hdf ..\myTarget.hdf

wam_remap_lla is a utility to remap a set of HDF files with external latitude/longitude array (LLA) to another projection. The LLA projection is quite versatile but very inconvenient and slow to use. Therefore it makes sense to remap those files to another projection. The syntax has 2 options.
In the first option *wam_remap_lla* reads a list of files to be remapped, a target filename (with the target projection), does the remapping and saves the remapped files. Either forward or inverse mapping procedure is used as selected in WIM *Settings-Special*. The list file should have 2 filenames in each line: the first name is the file with image to be remapped and the second filename (separated with a comma) is the filename with the LLA. Lines starting with # are skipped. It is advised to start with a small set of files, and you can mark the files to be skipped with the # character. The source image file may have many images (SDS-s), only one (fixed in the code) is read and remapped. A long list of attributes (e.g. Start Year, Start Day, End Year, End Day) are copied from the source images to the remapped images. This is done to facilitate the use of the remapped images in time series analysis.
The 2nd option is useful if the same LLA file is used for all the image files. You can use the same LLA filename in the list but it is more convenient to specify the LLA file in the command.

wam_remap_and_overlay

```
wam_remap_and_overlay vers. 2.6
Usage: wam_remap_and_overlay Pattern Overlay
Pattern is the matching filename pattern
Overlay is the hdf file with target projection and overlay
  (if Overlay is a RGB image then all images will be converted to RGB)
Options:
ConvertToByte=Yes converts float32 Chla to logScaled Chla
Lut=file.lut sets external LUT file
Median=M sets median filter with window size, e.g. 3
ColorMin=min and ColorMax=max set pixel values of color stretching
xPos=X sets x position of annotation, yPos=Y is y position of annotation
SavePng=No turns off saving PNG, default is to save PNG
Example:
  wam_remap_and_overlay C:\Sat\*.hdf C:\Maps\OL.hdf Median=3 xPos=100 yPos=10
```

Used to remap a whole series of HDF files to another projection and put a standard overlay on top of them. The Overlay file is used for both the target projection and as an overlay. The output is saved as both HDF and PNG. If the Overlay file is a WIM RGB image then the output images are also converted to RGB. Using RGB image allows using colors from different palettes. For example, you may want to make land gray but if the gray color is not available in the palette (e.g. in a typical palette like *chl1_white_end.lut*) then you can create a RGB Overlay image with gray land and use it as overlay. Remember that pixel value 0 and the black color in the RGB overlay image is considered transparent.

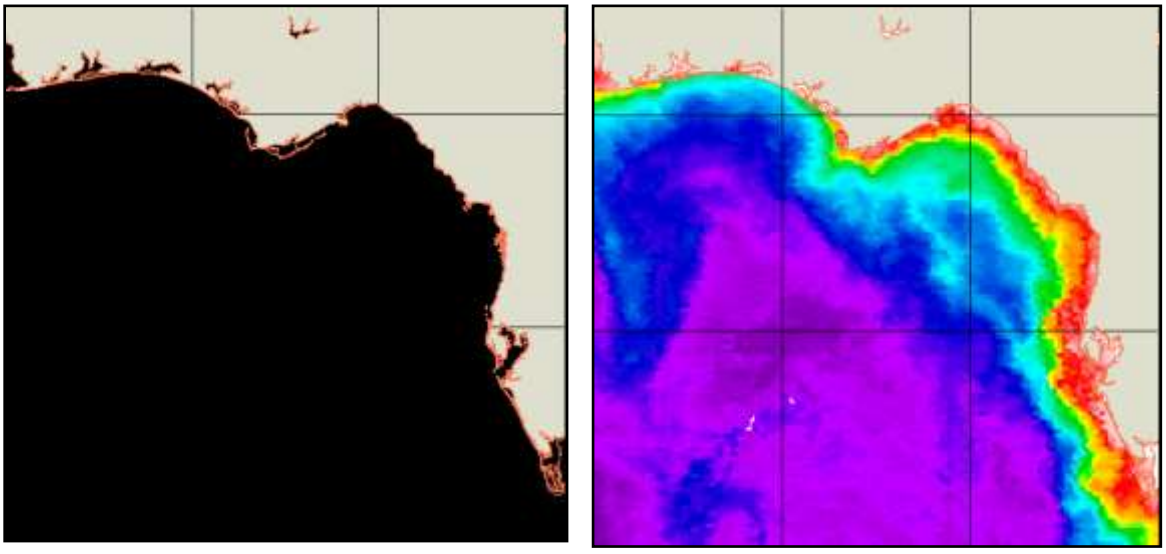
Instructions for making a RGB overlay with fixed RGB colors. Pick a projection image, create coastlines with *Geo - Get Map Overlay*, *coast_full.b*, *Background Value* = 0, *Foreground Value* = 1. Convert the image into RGB with *Transf-Convert to 24bpp (RGB)*. Fill the land areas manually with a color of choice, e.g. gray. Save as HDF. That will be your RGB overlay. You can also put gridlines, stations, color bars, etc on the overlay image.

A more difficult approach is to directly edit the LUT file. Instructions are given below but normally that is not needed. You can modify your favorite palette (LUT) file (**.lut*) with a text editor. Pick colors for coastlines (e.g. RGB = 240, 120, 85) and for land (e.g. RGB = 223, 223,

204). Edit the LUT file and change the pixel value 1 to 1, 1, 1, pixel value 2 to your coastline value and pixel value 3 to your land value. The beginning of the LUT file should look something like that:

```
0  0    0    0
1  1    1    1
2  240  120  85
3  223  223  204
```

Now fill land with pixel value 3 (with *Edit – Draw - Fill*, *Outline* = 3, *Fill* = 3). Load your modified LUT (*File – Look up Table – Load LUT*). If you want to add black latitude-longitude grid lines, do *Geo – Grid* with pixel value 1 (nearly black). Convert to RGB with *Transf – Convert to 24bpp (RGB)*. Save as HDF. This file will now be your *TargetProjection* (and overlay). A sample RGB overlay is shown below on the left and a sample output (overlaid) in the right. The purpose of these operations is to have land and coastlines with fixed RGB colors. If that is not required then the modification of the LUT and the conversion to RGB are not necessary.



wam_remap_to_kmz

Usage: wam_remap_to_kmz Pattern OverlayFile [xPos yPos [lutFile [Min Max]]]
  Pattern is the matching filename pattern,
  OverlayFile is the hdf file with target projection and overlay,
  xPos is annotation x position, yPos is annotation y position
  lutFile is external LUT file, Min and Max are respectively
    the pixel values of color stretching

wam_remap_to_kmz remaps, overlays and annotates a series of images with a date-time string. Output is saved as KMZ for visualization in Google Earth. The target projection has to be *Linear* (Google Earth requirement).

wam_remap_regions

Usage: wam_remap_regions Pattern

where Pattern is a matching filename pattern,
Target Projections and Overlays as well as
output locations are fixed in the code
Output is converted to RGB before overlaying.

Similar to *wam_remap_and_overlay* but works with multiple regions: it creates a number of remapped and overlaid images for each source image it reads. The regions and their projections as well as the output folders are fixed in the source code. Therefore you would need to adapt to your needs and recompile.

wam_xy2ll
Usage: wam_xy2ll FileName x y
If no x, y is given, calculates lat-lon for the Upper-Left and Lower-Right corners of the image

Calculates latitude and longitude from x, y.

wam_ll2xy
Usage: wam_ll2xy FileName Lat Lon
If no Lat and Lon is given, shows the size of the image
Calculates x, y from latitude and longitude.

3.2.8 Primary Production and Export Flux

wam_npp

wam_npp vers. 2.20
Usage: wam_npp PathsFile [Algorithm=Algorithm AnyYear=AnyYear LatLon=yes sds1=X sds2=Y sds3=Z]
Available Algorithms:
CbPM = Behrenfeld et al, 2005 Carbon based Productivity Model
DIERRSEN = Dierssen et al., 2000 Southern Ocean algorithm
ESQRT = Eppley Square Root of Chl
KI = Kameda & Ishizaka (2005) version of VGPM
Marra = Marra, Ho, Trees, 2003 model
SPGANT - Southern Ocean version of VGPM; modified Pbopt and Euphotic depth
50m - SPGANT with fixed 50 m Euphotic zone depth
SPGANTZEURRS - Southern Ocean version of VGPM using Euphotic depth from Rrs490/Rrs555
VGPM (Behrenfeld & Falkowski, 1997)
VGPM-Eppley - VGPM with Eppley (1972) PbOpt = f(SST)
VGPM-CAL - VGPM adjusted for CalCOFI data by Kahru et al., 2009
Default Algorithm is VGPM (Behrenfeld & Falkowski, 1997)
Option AnyYear must be a string of 0s or 1s with 1 in the Nth
position meaning that the Nth image can be from any year, e.g.
a composite over many years;
0 in the Nth position means to use year and day for finding Nth image
Default is to calculate difference in Days considering Years
Option LatLon=yes forces to save output as HDF with the Lat/Lon arrays - used for L2 data
Options sds1=X, sds2=Y, sds3=Z,... specify the sequence number of the SDS to read; default is 0

Example:
011 for VGPN would use EXACT year for Chl and ANY year for PAR and SST;
00101 for CbPM would use EXACT year for Chl/aph, bbp, K490 and ANY year for PAR and MLD.
PathsFile has paths and matching name pattern for all the images used
A sample PathsFile named paths.txt for VGPM (paths for Chl, PAR and SST):
C:\Sat\SEAWIFS\L3\Month\CHLO_9,S2006001*
C:\Sat\SEAWIFS\L3\Month\PAR_9,S2006001*
C:\Sat\MODISA\L3\Month\SST_9,A2006001*screened.hdf
A sample command using paths.txt:
wam_npp paths.txt Algorithm=VGPM-CAL AnyYear=100

Whereas *wam_npp_list* uses a detailed text file that specifies all filenames to be used, the list for *wam_npp* has just the directory names and the matching pattern and *wam_npp* itself finds the best matching set of images. The specific set of required images depends on the algorithm and typically includes *Chl-a*, *PAR* and *SST* for the VGPM algorithms and *aph* or *Chl-a*, *bbp*, *PAR*, *K490* and *MLD* for the CbPM algorithm. A sample list file *list_npp.txt* is provided. It uses data files in the *Images* folder of the WIM/WAM CD and assumes that you have copied the *Images* folder to *C:*. You can test *wam_npp*, with the following command:

wam_npp list_npp.txt

The list file list_npp.txt for VGPM algorithm has the following content:

```
# path for Chl files, matching pattern
C:\Images\SeaWiFS\baja_2000_april\,S200010*.hdf
# path for PAR files, matching pattern
C:\Images\SeaWiFS\L3\Month,*PAR*
# path for SST files, matching pattern
C:\Images\SST,2000*screened.hdf
```

For each matching Chl file *wam_npp* finds the best matching *PAR* and *SST* file using the files with minimal time difference with the Chl image. The maximum time difference is 30 days. Only the Kameda-Ishizaka option can be selected with *wam_npp* whereas *wam_npp_list* allows selecting all 3 options individually.

A sample list file for the CbPM algorithm (use the CbPM option in the command line) has the following content:

```
E:\sat\SEAWIFS\L3\Month\GSMchl,2006001
E:\sat\SEAWIFS\L3\Month\GSMbbp,2006001
E:\sat\SEAWIFS\L3\Month\PAR_9,S2006001
E:\sat\SEAWIFS\L3\Month\K490_9,S2006001
F:\sat\MLD\MLD,2006001
```

The paths specify, respectively, directories of images of *aph* or *Chl-a*, *bbp*, *PAR*, *K490* and *MLD*. The second argument on each line (after the comma) is optional and is used to match only specific matching files. In this example the January of 2006 (matching string 2006001) files are used.

wam_npp_lee

wam_npp_lee vers. 1.8
Usage: wam_npp PatternOfQAA PatternOfPAR [AnyYear]

Needs the following datasets in each QAA file: a490, aph440, bbp490
Expecting PAR in separate files as the first or only dataset
Extra option AnyYear must be a string of 0 or 1
 1 in the Nth position means that the Nth image can be from any year, e.g.
  a composite over many years;
 0 in the Nth position means that for Nth image years and days are considered
  when calculating the difference in days.
 Default is to calculate difference in Days considering Years
Example:
 0001 would use EXACT year for a490, aph440, bbp490 but ANY year for PAR

wam_npp_lee calculates images of Net Primary Production (NPP) using the Lee et al (2011) absorption-based AbsPP model ("An assessment of optical properties and primary production derived from remote sensing in the Southern Ocean (SO GasEx)", *J. Geophys. Res*., 116). Input datasets are *a490*, *aph440*, *bbp490* from a QAA-file (output from *wam_qaa_l2* or *wam_qaa_l3*) and PAR from a separate *PAR* file (e.g. standard mapped image from a Level-3 file). The PAR file can be from a different sensor, e.g. SeaWiFS or MODIS and will be remapped to the input images if the projection is different. The required input IOP variables (*a490*, *aph440*, *bbp490*) are automatically produced in *wam_qaa_l2*. When using Level-3 files in *wam_qaa_l3* you have to specifically select those variables.

Example command for SeaWiFS IOP data using SeaWiFS PAR:

```
wam_npp_lee    S1999\S1999*.hdf  PAR\S1999*.hdf
```

Example command for MERIS IOP data using MODISA PAR:

```
wam_npp_lee MER*mapped.hdf E:\PAR_9\2011\A2011016.L3m_DAY_PAR_par_9km
```

wam_npp_list

```
wam_npp_list vers. 1.2
Usage: wam_pp list_file [OPTION]
   OPTION can be KI
```

wam_npp_list calculates images of Net Primary Production (NPP) using the Behrenfeld and Falkowski (1997) VGPM model. Note that this is an older version and more convenient is to use the related *wam_npp* command.Various options are implemented. It needs a detailed list file specifying the directories where the input files are and all the individual filenames. Another, more flexible program *wam_npp* (see below) needs just the directory names and finds itself the best matching image files.
Sample list files are provided. A sample list file *list_calc_npp.txt* is for *wam_npp_list* and *list_npp.txt* is the list file for *wam_npp*. Both list files use data files in the *Images* folder of the WIM/WAM CD and assume that you have copied the *Images* folder to *C:\Program Files\Wimsof*t.
With the ("KI") option the Kameda and Ishizaka (2006) version of the VGPM is used.
In both lists the first 3 lines specify the paths of the chlorophyll, PAR and SST files, respectively. The list file for *wam_npp* can also use a string for finding matching files. The list for *wam_npp_list* must have lines with filenames of the Chl, PAR, and SST data, followed by the Julian day of the middle of the period, and optionally the three option parameters (see the WIM manual *Wim.pdf* for an explanation). The lines that start with the "#" character are skipped and

can be used for comments, to record alternative pathnames, etc. That is also a convenient way to mark sets of images that you do not want to use in the current run but would like to keep in the list. The NPP images are created of type Int16 and include the various parameter values as HDF attributes.
In summary, in order to calculate primary production images (either global or regional), set up your Chl, PAR and SST images in specified folders. Then create a simple text file, with a list of images and options to be used. You can use a provided list file *list_cal_npp.txt* as a template. To test the program, run

wam_npp_list list_calc_npp.txt

Note! You have to remove the "Read-only" property of the file (right-click-Properties).
The function that calculates primary production has 3 optional parameters that follow the list of image names. The default values are 1,1,1, i.e. each line is followed by three values of one. The first parameter specifies *photoinhibition* (2 = no *photoinhibition*, 1 and anything else = *photoinhibition*). The second parameter specifies *PBopt* functions (see WIM manual). The third parameter specifies euphotic zone depth calculation (2 = Morel and Maritorena, 2001 model; 3 = SPG Southern Ocean formulation; 1 and anything else=Morel and Berthon, 1988).

wam_npp_point

```
wam_npp_point vers. 2.18
Usage: wam_npp_point ListFile [Algorithm] [AnyYear]
Available Algorithms:
  CbPM = Behrenfeld et al, 2005 Carbon based Productivity Model
  DIERRSEN = Dierssen et al., 2000 Southern Ocean algorithm
  ESQRT = Eppley Square Root of Chl
  KI = Kameda & Ishizaka (2005) version of VGPM
  Marra = Marra, Ho, Trees, 2003 model
  SPGANT - Southern Ocean version of VGPM; modified Pbopt and Euphotic depth
  50m - SPGANT with fixed 50 m Euphotic zone depth
  VGPM (Behrenfeld & Falkowski, 1997)
  VGPM-Eppley - VGPM with Eppley (1972) PbOpt = f(SST)
  VGPM-CAL - VGPM adjusted for CalCOFI data by Kahru et al., 2009
Default (missing) is VGPM (Behrenfeld & Falkowski, 1997)
Extra option (argument2, AnyYear) must be a string of 0 or 1
  1 in the Nth position means that the Nth image can be from any year, e.g.
    a composite over many years;
  0 in the Nth position means that for Nth image years and days are considered
    when calculating the difference in days.
  Default is to calculate difference in Days considering Years
Example:
  011 for VGPN would use EXACT year for Chl and ANY year for PAR and SST;
  00101 for CbPM would use EXACT year for Chl/aph, bbp, K490 and
ANY year for PAR and MLD.
ListFile has paths to the matching satellite data and a list of points (stations
)
A sample ListFile:
C:\Images\SeaWiFS\baja_2000_april\S2000*.hdf
C:\Images\SeaWiFS\L3\8day\S2000*PAR
```

C:\Images\SST*screened.hdf
C:\Images\match.csv

wam_npp_point calculates net primary production (NPP) for a list of "stations" specified by the longitude, latitude, date and time and using series of images. NPP is calculated per day (mg C/m^2/day), i.e. the time value is actually not used and is used only for compatibility with the *wam_match* program.

Note the difference with other programs in this section: whereas *wam_npp_list, wam_npp* and *wam_ef* calculate new images, *wam_npp_point* calculates NPP values and saves the statistics as text in a spreadsheet type CSV file.

The available algorithms include versions of the Behrenfeld and Falkowski (1997) VGPM model, the Kameda and Ishizaka (2005), the Behrenfeld at al (2005) Carbon based Productivity Model (CbPM) and the simple Eppley square root of Chl (ESQRT) (Eppley et al., 1985). The VGPM algorithm assumes all default options (corresponding to options 1,1,1 in *wam_npp_list*). Other options and algorithms may be added.

The idea behind *wam_npp_point* is to give paths of satellite files and let it find the best matching files and extract the pixel data for each particular neighborhood (station). A text file with paths of satellite data and the name of the list of stations is used as input. The list file needs to have 4 or more lines specifying, respectively, the three paths of the Chl, PAR, and SST files, and (the 4th line) the filename with the list of stations. The lines starting with the # character are skipped. You can use the # character to comment out lines that you may want to use later. You should also specify a matching string of characters that is used to match the files in that directory. For example, Chl and PAR files may be in the same folder and then it is essential to keep them separate with the matching strings of *PAR* for PAR files and *CHL* for Chl files. The matching string is separated from the path name with a comma. Note that the **subdirectories** under the given directory are also searched for matching files. Therefore, you can have your files in separate subdirectories for different years, e.g. "1999", "2000", "2001", "2002", etc and the matching files will still be found. A sample list file *list_npp_point.txt* is included in the *Wimsoft* folder. In that file I have commented out the "real" paths that I used in a real project and I am using sample paths of the images that are included on the WIM CD in folder *Images*. I assume that you have copied the *Images* folder from the WIM CD to your *C:\Images*.

The list file *list_npp_point.txt* has the following contents:

```
#E:\sat\SEAWIFS\L3\Daily\CHLO_9\*CHLO_9
#E:\sat\SEAWIFS\L3\Daily\PAR_9\*PAR_9
#F:\SST\MODISA_8day\SST_9\A*screened.hdf
C:\Images\SeaWiFS\baja_2000_april\S2000*.hdf
C:\Images\SeaWiFS\L3\8day\S2000*PAR
C:\Images\SST\*screened.hdf
C:\Images\match.csv
```

The list of stations (*match.csv* in this example) is the typical comma or tab separated (CSV) file with the following columns: *Longitude*, *Latitude*, *Date*, *Time* and any number of additional columns. It is also called the WIM "point" file and the format must be carefully followed, e.g. longitude and latitude must be decimal numbers, the date and time must be either in the US format (e.g. MM/DD/YYYY). Please see "***List of Point data***" for the ***wam_match*** program in

this manual for more details. In this example I am using the same sample file (*match.csv*) that is used in the *wam_match* example. Please note that although the *Time* column in the *match.csv* file is not used in *wam_npp_point*, it must be present in the file and have some data. You can fill it with any dummy time data, e.g. 12:00.

wam_npp_point reads all the matching filenames for each image type. For each station (point) in the point file the image files are sorted by the closeness in time to the current point. In the best matching image a 3 x 3 pixel window is extracted centered at the nearest pixel. It is common that the best matching image in time is cloudy over the particular station. It is required that at least 3 pixels out of the 9 are valid for each of the component images. If less than 3 valid pixels are found then the next matching image is used. This switching to the next matching image is repeated until enough valid data is found or the maximum difference between a station (point) and an image (30 days) is reached. For all valid pixels in the 3 x 3 pixel neighborhood NPP is calculated and the statistics is saved to a file. The output filename is constructed from the list file name and the algorithm name. It has the extension ".csv" and is saved in the current directory. You can test it by issuing the following command:

```
wam_npp_point list_npp_point.txt
```

Each line in the output file has a copy of the values in the columns of the input file and adds the following columns:

DiffDays1, DiffDays2, DiffDays3 (the difference in days between the station and the used satellite image for Chl, PAR and SST, respectively), *Image* (the Chl, PAR and SST image names "glued" together with the + sign), *SYear* (start year), *EYear* (end year), *SDay* (start day), *EDay* (end day), *Station* (station name), *Nin* (number of valid pixels), *Nout* (number of invalid pixels), *Min* (minimum of NPP), *Max* (maximum of NPP), *Mean* (mean of NPP), *StDev* (standard deviation of NPP), *Median* (median of NPP), *Pointvalue* (NPP from the nearest pixel, i.e. the center of the 3 x 3 pixel window), *Pixelvalues* (9 NPP values from the 3 x 3 pixel window). Invalid NPP values due to invalid Chl, PAR or SST have -99 value. Lines with no NPP data are skipped. The purpose of copying the input data into the output file is to make it easy to compare satellite data with in situ data.

Whereas standard PAR data are currently available only from Level-3, more options are available for picking satellite Chl and SST data. You can choose between different sensors, different algorithms, different spatial resolutions and different compositing intervals (e.g. Level-2, Level-3 daily, 8-day, monthly). The current version of *wam_npp_point* assumes that Chl, PAR and SST data are always in the first SDS (as in L3 data). Therefore L2 data cannot be used in general. However, you can extract the Chl data from Level-2 files with *wam_series* and save as separate Chl images, possibly remapping to a standard map.

Here are some considerations for picking the satellite data. The purpose of *wam_npp_point* is to provide best matching satellite NPP values for in situ measurements. In situ measurements are normally conducted over a period of time that is less than 1 day. Therefore, the best set of matching images are the daily Chl, daily PAR and daily SST. Daily Chl and SST images are often cloudy or have no orbit coverage. PAR images are not affected by clouds but are affected by orbit coverage (not all days have coverage). If there is no data in the best matchup image, i.e. in the image with the least time difference between a station (point) and a satellite image then *wam_npp_point* will pick the next closest image until it reaches the last image or 30 days of maximum difference. You can also use 8-day images or even monthly images. When using monthly images the current month must have valid data as the next month is always beyond the maximum time difference of 30 days. As Chl has the most influence on NPP it is important to use the best Chl matchup. The influence of SST is relatively weak and therefore it may be a good idea to use interpolated (e.g. *bsst* data for AVHRR Pathfinder) or composites over a longer period (e.g.

5-8-day) in order to not lose any matchups due to no SST data. Multiple data formats (e.g. Pathfinder v5 SST and OCPG SST) can be used for SST. The Pathfinder BSST data has interpolated SST and has no missing pixels due to clouds. This is probably the best SST to use in NPP calculations as it guarantees a reasonable if not the most accurate SST value.

The statistics is calculated based on all valid 9 pixels in the 3 x 3 pixel window. The current version of *wam_npp_point* finds the center of the 3 x 3 pixel window by the *Longitude* and *Latitude* of the point but it does not remap the other neighboring pixels. Therefore, if the spatial resolution or the projection of the Chl, PAR and SST images are different then the pixels other than the center pixel may not be exactly matched.

The sample list file *list_npp_point.txt* uses remapped Level-2 Chl images in the *SeaWiFS\baja_2000_april* folder, global Level-3 8-day PAR and SST composites. If you don't have these sample images then you can easily use other Level-3 images. Keep in mind that *wam_npp_point* has to find the best matching images for each station (point) using the file name and it may be confused if you have different types of images in the same folder. Therefore it is important to keep different data files in separate directories and to match only the correct files using the matching string. For example, *S2000*PAR* is the matching string for PAR images used in the example.

wam_ef

```
wam_ef vers. 2.1
Usage: wam_ef PathsFile [Fraction] [AnyYear] [Dunne/Laws2004/Laws2011/Maiti]
Dunne (2005), Laws (2004), Laws (2011) and Maiti (2013) are the alternative
models; default is the Laws (2011) model
Laws2004 and Dunne2005 need 3 image paths (NPP, SST, Chl),
Laws2011 needs 2 (NPP, SST) and Maiti2013 needs 1 (NPP)
PathsFile has paths and matching file name pattern for, e.g., NPP, SST and Chl
A sample PathsFile using NPP, SST and Chl:
  C:\Sat\SeaWiFS\L4\Monthly\NPP, S2001*.hdf
  C:\Sat\Merged\L3\Monthly\SST_9, M2001*screened.hdf
  C:\Sat\SEAWiFS\L3\Monthly\CHLO_9, S2001*
Laws2004 ef is multiplied by 0.725, an empirical correction factor, to make it
  more realistic
Laws2011 model does not need Chl and uses only NPP and SST
Maiti2013 model does not need SST and Chl and uses only NPP
Optional argument Fraction saves the fraction EF/NPP in a separate HDF file
Any other string in its place means no fraction saving.
Will match each NPP image with the closest matching SST and Chl image
  using the Year and YearDay of each image.
Extra option AnyYear must be a string of 0s and 1s with the purpose of using
  climatological means for some variables, specified by 1 in the AnyYear string.

The sequence of variables in the AnyYear string is also NPP, SST, Chl.
1 in the Nth position means that the Nth variable can be from any year, e.g.
  a composite over many years;
0 in the Nth position means that for Nth variable years and days are considered
Default is 000, i.e. to consider both Years and Days when matching images.
Example:  010 would use EXACT years for NPP and Chl but ANY year for SST.
```

wam-ef will match each NPP image with the closest matching SST and Chl image using the Year and *YearDay* of each image.
Extra option *AnyYear* must be a string of 0s and 1s with the purpose of using climatological means for some variables, specified by 1 in the AnyYear string.
The sequence of variables in the *AnyYear* string is also NPP, SST, Chl.
1 in the Nth position means that the Nth variable can be from any year, e.g. a composite over many years;
0 in the Nth position means that for Nth variable years and days are considered. Default is 000, i.e. to consider both Years and Days when matching images.
Example: 010 would use EXACT years for NPP and Chl but ANY year for SST.

wam_ef uses various models to calculate export flux. The following models are currently implemented: Dunne2005 (Dunne et al, *Global Biogeochemical Cycles*, vol. 19, GB4026, doi:10.1029/2004GB002390, 2005), Laws2004 (Laws, E.A., *Prog. Oceanogr*. 60: 343–354, 2004) and Laws2011 (Laws, E.A., et al. *Limnology Oceanography Methods* 9: 593-601, 2011), Maiti2013 (Maiti et al, *Geophyscal Research Letters*, vol. 40, 1-5, doi:10.1002/grl.50219, 2013). The Laws2004 and Dunne models use primary production, SST and surface Chl-a (to calculate euphotic zone depth according to a certain parameterization, e.g. Morel and Maritorena, 2001). Laws2011 uses NPP and SST, Maiti2012 uses only NPP. The image paths and matching strings must be in list of paths and matching filenames, e.g. for NPP, SST and Chl:

```
# PATHS for NPP, SST, Chlor_a
F:\SeaWIFS\L4\Monthly\NPP_SPGANT_blended, S2001*.hdf
F:\AVHRR\Month\pf5_month_asc\bsst, 2001*.hdf
#F:\Merged\L3\Monthly\SST_9, M2001*.hdf
F:\SeaWIFS\L3\Monthly\CHLSPGANT_blended, S2001*.hdf
```

The output is Export Flux (*EF = ef * NPP*) where *ef* is the export fraction and NPP is net primary production. EF is saved in Int16 format. Other details, e.g. units, input files, etc. are recorded in the HDF attributes. If the optional argument *Fraction* is used then the export fraction of PP (ef) is saved as a separate byte image.

3.2.9 Statistics

wam_fill

```
wam_fill  vers. 2.10
Usage: wam_fill Pattern [nRun [dX [dY [Mask [PatternIce]]]]]
Pattern is pattern of matching HDF filenames
nRun is the number of iterations, dX and dY are the window size in pixels.
Defaults: nRun = 2, dX = 3, dY = 3
Mask is a HDF image with valid area masked with value 1,
e.g. ocean mask must have pixel value 1, everything else is made invalid
PatternIce is a pattern of HDF filenames of ice concentration (scaled byte)
If ice pixel value > 0, the filled pixel will be made invalid.
```

wam_fill fills missing (invalid) pixels with the mean pixel value of the dX x dY window around the pixel. HDF files matching *Pattern* are used.
A common problem when filling invalid values is that the filling process may cover land (e.g. small islands or other coastline) where we do not want to fill with valid pixels. The purpose of

Mask is to prevent filling islands and other land next to coastline. The *Mask* image specifies valid ocean area with pixel value 1 and all pixels outside the valid area will be masked with pixel value 0 or 255. The *Mask* specifies a constant area where filling is expected. A more complex situation is presented by ice cover that is not constant but variable. As ice cover is changing daily and seasonally, multiple ice images are needed to specify are covered by ice. When the matching pattern of ice images *PatternIce* is provided, *wam_fill* searches all matching ice images and finds the closest matching ice image. The ice images need to be of pixel type *Byte* whereas the filled images can be either *Int16* or *Byte* type. All pixels in the matching ice image with pixel value > 0 are considered ice or land and the corresponding pixels in the filled image will be not be filled (they are set to either 0 or 255). The mask and the ice images must have the same dimensions as the images to be filled. An example using ice images:

```
wam_fill  *.hdf   2  5  5   ..\ocean.hdf   ..\ice\nasateam_final_mo_1440x720\nt_*.hdf
```

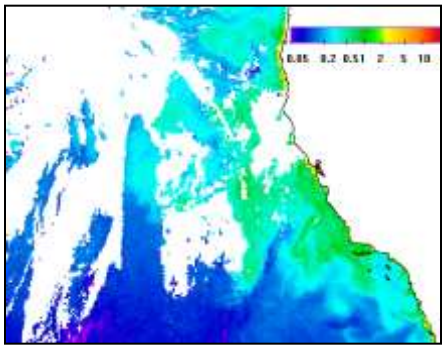
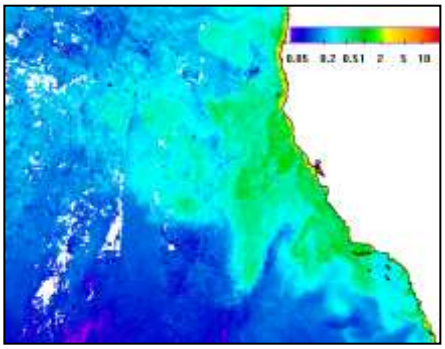
wam_fill_intime

```
wam_fill_intime  vers. 1.1
Usage: wam_fill_intime Pattern [nLength]
Pattern is pattern of matching HDF filenames
nLength is the number of images in time that are used in the filling,
  default is 3. Currently it is fixed at 3 and cannot be changed!
```

wam_fill_intime fills missing (invalid) pixels by interpolating between the same pixel value in previous and next images or replacing the invalid pixel with a valid pixel value from the next image (if previous value was invalid). You can also run it twice to fill even more invalid pixels. The first and the last image in the sequence are not being filled (interpolated into), therefore, when running for 2nd time, copy the first and the last image to the folder with the 1-time interpolated images.
An example command below interpolates for all matching hdf files in the SST_4 folder::

```
wam_fill_intime SST_4\*.hdf 3
```

A typical image before (left) and after (right) running *wam_fill_intime* is shown below:

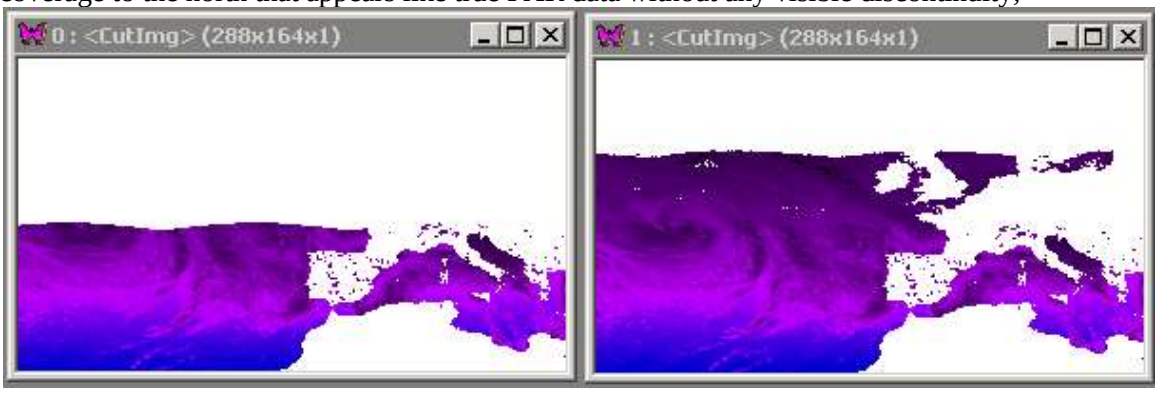



wam_fill_with

```
wam_fill_with  vers. 1.1
Usage: wam_fill_with Pattern PatternToFill  [Options]
Pattern is pattern of matching filenames (HDF4)
PatternToFill is pattern of matching filenames (HDF4) for filling
Options:
```

mask=MaskImage specifies the mask to consider (pixel value 1)
isSIS=yes converts from SIS to PAR before filling
anyYear=yes does not consider year when matching files by time

wam_fill_with fills missing (invalid) pixels with corresponding pixel values from another dataset. The matching datasets are found by closeness in time: the closest dataset is time will be used for filling. A separate option (isSIS=yes) is designed for filling missing pixels in PAR images with corresponding datasets of SIS (Surface Incoming Shortwave irradiance) with a linear relationship: PAR = 0.1765 * SIS – 0.789. Another option anyYear=yes is used to fill with seasonally close data irrespective of the actual year.
For example, PAR data from ocean color sensors has a limited range at high latitudes due to a threshold of sun zenith angle. In practice, PAR and other ocean color data are missing in winter at higher than ~50N or 50S although there is still significant light. The datasets of surface incoming shortwave (SIS) flux derived from geostationary or weather sensors (like AVHRR) have data at higher latitudes that can be used to fill in the missing pixels if corresponding SIS data is available. The figure below shows the ocean color PAR image (left) from January 1, 2009 and the same data after filling missing pixels with PAR data converted from SIS. You can see the extended coverage to the north that appears like true PAR data without any visible discontinuity,



An example command that produced the filled image in the right is:

```
wam_fill_with  PAR_25\*.hdf SIS\*.hdf mask= 1440x720_ocean.hdf  isSIS=yes
```

wam_histogram

```
wam_histogram vers. 1.0
Usage: wam_histogram PATTERN Mask
  where PATTERN is a matching pattern of hdf filenames
    Mask is a HDF file with pixel value = 0 for no mask
    and pixel value != 0 for the mask
If no Mask file is given, the whole image is used
```

Calculates histograms of pixel values for a set of 1 byte-per-pixel images in HDF files, saves the histogram in a CSV file. Currently the images have to be 1 byte-per-pixel and the histogram bins (brackets) are simply the pixel values from 0 to 255. You can specify a sub-area of the image where you calculate the histograms with a mask file. The mask image must have the same size as the images used in the histogram counts. Any pixel value different form 0 means that the pixel is part of the mask where histograms are counted. The output is saved as a CSV file with the image file in the first column and the counts corresponding to pixel values in the following 255 columns. The output file can be easily read into Excel for further analysis.

wam_integrate

```
wam_integrate vers. 1.5
Usage: wam_integrate Pattern [DeltaLon DeltaLat [MaskImage]]
  where Pattern is a matching pattern of HDF filenames,
  DeltaLon and DeltaLat are the size of the window,
  MaskImage is HDF file that specifies region of interest with nonzero pixels.
Mask values can be: 1, 2, 3, ... or just 1 for a single area of interest.
Mask values MUST be consecutive.
Defaults are 1 deg Latitude and 1 deg Longitude for Deltas
Example: wam_integrate *.hdf 1 1 OceanMask.hdf
Here a sample mask OceanMask.hdf specifies our area of interest.
```

wam_integrate runs through a set of matching standard mapped images (in global equirectangular or equal angle projection) and performs spatial integration. For example:

```
wam_integrate SeaWIFS\L4\Monthly\S1998*.hdf  1  1 Satmasks\OceanMask.hdf
```

is using matching primary production files *S1998*.hdf* to calculate global primary production (in Gigatons of Carbon per month, the same as Petagrams of Carbon per month) for the ocean area specified in the mask image in *Oceanmask.hdf*. By using a mask image it is possible to calculate the integrals in more than one specified area, e.g. separately for different latitude bands. A major problem with most satellite datasets is that some pixels have no data due to clouds or other reasons and these missing pixels have to be dealt with in some way. The *DeltaLon* and *DeltaLat* arguments specify the size of a small window in degrees of longitude and latitude where the mean value is calculated based on all valid pixels in the window and the mean value is assumed for all the missing pixels in that window. A problem with that approach is that we may assign the mean value to pixels in that window that represent land or ice. To prevent that from happening we use a *Mask* image that specifies all possible valid pixels. The *Mask* image also allows calculating separate integrals for more than one area. The *Mask* image must specify areas of interest with consecutive pixel values starting from 1. The *Mask* image must have the same size as all the images to be used and be a byte image with pixels from area(s) of interest having values starting from 1 and area to be excluded have pixel value of 0.
It is crucial that only the valid values are used in the calculation, i.e. *wam_integrate* must guess the valid range of values and this is reported for the 1st image. For example, output of

```
S19980011998031.L3m_MO_CHLSPGANT.hdf:    fMin=1, fMax=30000
```

reports that the valid range is from 1 (*fMin*) to 30000 (*fMax*). The valid range is retrieved for each image but is shown only for the 1st image.

wam_map_match

```
wam_map_match vers. 1.2
Usage: wam_map_match MatchFile PathsForImageFiles MapOverlay
  where MatchFile is output from matching, i.e. a CSV text file with Lon,Lat,..
  PathsForImageFiles is path for image files,
  MapOverlay is HDF4 file of the map that will also be overlaid
Option: Annotation=no, default is to have the O annotation on the matchup
Example:
```

wam_map_match match_VIIRS.csv D:\CAL\2012\V2012\0\Good maps\myMap.hdf

C:\Windows\System32>wam_map_match
wam_map_match vers. 1.2
Usage: wam_map_match MatchFile PathsForImageFiles MapOverlay
 where MatchFile is output from matching, i.e. a CSV text file with Lon,Lat,..
 PathsForImageFiles is path for image files,
 MapOverlay is HDF4 file of the map that will also be overlaid
Option: Annotation=no, default is to have the O annotation on the matchup
Example:
wam_map_match match_VIIRS.csv D:\CAL\2012\V2012\0\Good maps\myMap.hdf
You may have your images in different directories, e.g. in directory D:\2012\V2012\0\Good and in directory G:\2013\V2013\0\Good; then you can just list these directories in the command line and it is assumed that the last name in the command line is the map and overlay file:

wam_map_match match_VIIRS.csv D:\2012\V2012\0\Good G:\2013\V2013\0\Good maps\myMap.hdf

Note that naming of the output HDF4 and PNG files includes appending a sequence number to the image file name. The numbering follows the sequence of matchups in the match-up file and starts with 002 (instead of 001), followed by 003, etc. This is designed to facilitate comparison with the match-up file that has been read into Excel or another editor or spreadsheet as line 1 is occupied by the Header. In each image the matchup is indicated by an "O" annotation. This annotation is written into the mage, i.e. it changes the pixel values. In case you don't want the "O" annotation, you can use the Annotation=no option in the command line and the "O" annotation will be written into the PNG file only and not into the HDF4 file.

wam_match_l2

wam_match_l2 vers. 2.36
Usage:
wam_match_l2 PointFile Pattern [variable1 variable2 ..]
 where PointFile is a CSV file with Lon,Lat,Date,Time,...
 Pattern is a matching pattern of L2 HDF filenames, e.g.
 E:\2015\A2015\A*.hdf
Default variables are chlor_a (algal_1 for MERIS) and l2_flags,
 additional variables can be specified
Options:
MaxDiffDays=D selects only those with time difference < D days. Default is 30.
GetTimeFromAttributes=Yes gets the scene center time from attributes;
 default is to use time from file name which is faster but reports time
 that is several minutes earlier (i.e. of the 1st scan and not center)
Typical example (note: chlor_a and l2_flags are included by default!):
wam_match_l2 Stations_2015.csv E:\2015\A2015\A*.hdf Rrs_443 Rrs_488 Rrs_547 Kd_4
90 maxDiffdays=5 sensor=MODISA gettimefromAttributes=yes

MaxCoeffOfVariation=X selects only those with between-pixel CV < X.
 Default is not to use this. CV = SD/Mean. Commonly used maxCoeffOfVariation
 is 0.4.
nValidFrom=X selects only those with at least X valid pixels. Default is 1.
MaxPixelsFromCenter=M limits the matchups to M pixels from the center of the

```
  scene, e.g. 530 corresponds approximately to HISATZEN for SeaWiFS MLAC scene
  and eliminates swath edges.
BadFlags=File.csv loads a list of 32 L2-flags that make a pixel invalid.
  File must be a CSV file with a header. The 1st column must be either 0 or 1;
  1=use that flag; 0=flag not used.
  2nd column may be a comment, e.g. flag name.
NoFlags=yes uses variable range instead of L2-flags for testing pixel validity
ShowFlags=yes shows the name of the first L2-flag that makes pixel invalid
Default flags used: ATMFAIL,LAND,HISATZEN,CLDICE,CHLFAIL,NAVFAIL

Advanced usage: can apply algorithms to matchup-values (not recommended here!)
wam_match_l2 PointFile Pattern [variable1 variable2 .. algorithm=ALGORITHM
  sensor=SENSOR]
  Variables chlor_a (algal_1 for MERIS) and l2_flags are always included
Available Algorithms are:
  OC2, OC3, OC4 (using latest NASA coefficients depending on sensor),
  SPGANT3 and SPGANT4 for the Southern Ocean (Mitchell & Kahru, 2009)
Only 1 sensor and 1 algorithm can be used at the same time
Possible Sensors are:
  SEAWIFS OCTS OCM1 OCM2 MOS MERIS MERISL2 MODIST MODISA HMODISA HMODI
ST CZCS OSMI VIIRS GLI MERGED MERISNASARR MERISNASAFRS UNKNOWN
Not all sensor and algorithm combinations are implemented
Examples:
wam_match_l2 Points.csv E:\A2012*.hdf Rrs_412 Rrs_443 maxDiffDays=5
wam_match_l2 Points.csv E:\A2009*.hdf BadFlags=C:\Data\MyFlags.csv
wam_match_l2 Points.csv E:\MER*.hdf algal_1 algal_2 maxDiffDays=5
```

wam_match_l2 is a command line utility that is similar to the GUI application *wam_match* but *wam_match_l2* has much more versatility. The main difference is that *wam_match_l2* finds only the match-up that is the **nearest in time** and satisfies other requirements. It reads a *Point* file (in CSV format) with station information in the form of *Longitude*, *Latitude*, *Date*, *Time* (in that sequence!) and other variables like *Cruise*, *Station*, *Value*. It then scans a list of matching Level-2 ocean color files and finds the nearest satellite image in time with sufficient valid data to create statistics for the 3 x 3 pixel window centered at the nearest pixel to the station. The time and date have to be in GMT (UTC) as satellite data are recorded using these. This program assumes that it is applied to Level-2 ocean color datasets that are provided by the NASA or ESA MERIS (MERIS Level-2 RR data converted to HDF4 with *wam_convert_n1*). The program sorts all images by their proximity to the date and time of each point (station). It then goes through the sorted list of images and tries to find valid (typically, cloud-free) data for each point (station). If the nearest dataset in time is not accepted (i.e. is cloudy or has no valid data for other reasons), it switches to the next nearest Level-2 dataset (file) in time going either back in time or forward in time. This process continues until the first valid match-up is found or until the maximum allowed time difference (30 days by default) is reached. It has many options. For the most up-to-date options type the name of the command.

The maximum time difference in days is set in the command (e.g. maxDiffDays=5). In contrast to *wam_match*, you don't need a fixed list file with image names to be used. Instead, it uses all matching files, e.g. *C:\Sat\MODISA\L2\A2008300*.hdf* (note the *).

Match-up has to pass several requirements, e.g. the number of valid pixels in the 3 x 3 pixel window is at least *nValidFrom* . The default value for *minNvalid* is 1 but that can be changed with the option *nValidFrom*=N. As the process of finding Level-2 match-ups is rather slow, it is a good idea to initially collect more match-ups and apply more stringent screening later with *wam_read_match*. Therefore, in the default case a 3 x 3 pixel area needs at only 1 valid pixel out of the 9 total pixels. *L2_flags* are used to define valid pixels. The ”bad” flags that make a pixel invalid for the NASA Level-2 data are: ATMFAIL, LAND, HISATZEN, CLDICE, CHLFAIL, NAVFAIL. Flag, PRODFAIL has been in this list but has been removed as of May, 2014. Again, it is a better idea not to eliminate too many match-ups initially but apply more targeted screening later with *wam_read_match*. For MERIS L2 files converted to HDF with *wam_convert_n1* the “bad” flags are: LOW SUN, HIGH_GLINT, ICE_HAZE, SUSPECT, COASTLINE, PCD_19, PCD_18, PCD_17, PCD_16, PCD_15, PCD_14, PCD_1_13, CLOUD, LAND. Option *showFlags* shows the first flag that makes the pixel invalid. Option *noFlags* skips using l2-flags and uses the pixel value range for testing validity. Variable *l2_flags* is still required to be in the file.

You can specify your own set of flags that make a pixel invalid for matchups with the option *BadFlags=File.csv* where *File.csv* is a CSV text file (e.g. *C:\Data\MyFlags.csv*) and has the following structure:

```
Value,  Name
    1,  ATMFAIL
    1,  LAND
    0,  PRODWARN
    0,  HIGLINT
    0,  HILT
    1,  HISATZEN
    0,  COASTZ
    0,  SPARE
    0,  STRAYLIGHT
    1,  CLDICE
    0,  COCCOLITH
    0,  TURBIDW
    0,  HISOLZEN
    0,  SPARE
    0,  LOWLW
    1,  CHLFAIL
    0,  NAVWARN
    0,  ABSAER
    0,  SPARE
    0,  MAXERITER
    0,  MODGLINT
    0,  CHLWARN
    0,  ATMWARN
    0,  SPARE
    1,  SEAICE
    1,  NAVFAIL
```

0, FILTER
0, SSTWARN
0, SSTFAIL
1, HIPOL
0, PRODFAIL
0, SPARE

Note that the flags file must have a header line that is followed by at least 32 lines. Only the first column is used and subsequent columns can be used for comments. Separator is typically a comma but can also be a tab, space or semicolon. Value 1 means that this flag will be used, i.e. if this flag is set, the pixel is considered invalid. Value 0 means that this flag will not be used, i.e. it does not matter if the flag is set or not. The flags specified above are the default flags that are used when the *BadFlags=File.csv* option is not applied. The meaning of the flags may depend on the sensor (e.g. different for MERIS) and may change in the future. One of the flags that is used by default is HISATZEN (i.e. high satellite zenith angle) which typically means that the pixel is at the edge of the swath and has low quality. It is also possible to eliminate the swath edge pixels with an option *Option maxPixelsFromCenter=M* limits matchups to *M* pixels from the center of the scene. For example, using option *maxPixelsFromCenter=530* eliminates all pixels outside of the center by more than 530 pixels. This corresponds approximately to the HISATZEN flag for SeaWiFS MLAC scene (zenith angle larger than 60 degrees) and eliminates matchups that are at swath edges.

In the simplest form *wam_match_l2* extracts only *chlor_a* and *l2_flags* datasets and does the statistics for a 3 x 3 pixel neighborhood centered at the nearest pixel. Only valid pixels defined by the flags are used in calculating the minimum, maximum, mean, standard deviation and median. All the *l2_flags* that are set in all the 9 pixels (3 x 3 pixel area) are recorded in the output file. The output is appended to the input data and includes the following:

StartYear,EndYear,StartDay,EndDay,Image,TimeDiff_Days,SceneCenterSolarZenith,VarName,Pointvalue,Nin,Nout,Min,Max,Mean,StDev,Median,VarName,Pixel1,Pixel2,Pixel3,Pixel4,Pixel5,Pixel6,Pixel7,Pixel8,Pixel9.

Note that statistics are given for each variable and the 9 pixel values of the flags finish the line.

TimeDiff_Days is the time difference between the satellite pass and the in situ station. It is negative if the satellite pass is before the in situ station and positive if it is after the station.

The timing of each satellite file is normally read from the file name. That time usually corresponds to the start of the scene. If the scene is big then the correct time in the center of the scene can be several minutes off. Normally a few minutes timing error is not significant but you can also use the option *GetTimeFromAttributes=yes*. That option forces the program to read each file and get timing from the attributes. The default attribute is *Scene Center Time* but other attributes can also be used if this is missing. Be warned: this option makes *wam_match_l2* very slow and is therefore not recommended!

The more advanced options allow picking specified variables (e.g. *Rrs* bands) and optionally applying specific algorithms to those in order to calculate alternative *Chl* or other outputs. For applying other algorithms w*am_match_l2* needs to extract values of the spectral bands of *Rrs* (remote sensing reflectance) or *reflec* (reflectance for MERIS). Note that previous versions of NASA ocean color files included *nLw* (normalized water-leaving radiances) instead of *Rrs*. As some algorithms still use *nLw* instead of *Rrs*, for those algorithms *Rrs* is converted to *nLw* by multiplying with the corresponding solar irradiance value that is read from an attribute in the file.

w*am_match_l2* then calculates statistics for each point using the selected algorithm. For example, when reading MODISA data, the standard algorithm is OC3 that was already used by NASA to calculate in the *chlor_a* values in the L2 file. You can select OC3 algorithm to calculate your own *Chl* values and they should approximately agree with the standard *chlor_a* values calculated by NASA. Minor differences are expected due to conversions between *double* and *float* number formats, etc. An advantage of *wam_match_l2* is that it allows testing various other algorithms to find match-ups without creating the corresponding image files. The list of available algorithms can be easily expanded. However, it is better idea to collect the match-ups first and apply the various algorithms later to the extracted match-up data.

A sample *Point* file is given below. You can exclude lines from being used by using the # character in the beginning of the line.

```
Lon,Lat,Date,Time,Station,Chl
-118.159,33.495,10/26/2008,21:45,1,0.368
#-118.105,33.434,10/26/2008,22:45,2,0.436
#-118.011,33.402,10/26/2008,23:45,3,0.623
-114.406,31.325,10/27/2008,15:45,4,2.408
```

Sample commands for, respectively, SeaWiFS, MODIS-Aqua and MERIS are:

```
wam_match_l2  Points.csv  C:\CAL\2009\S2009\S2009*.hdf  maxDiffDays=5  sensor=SEAWIFS
wam_match_l2  Points.csv  C:\CAL\2009\A2009\A2009*.hdf  maxDiffDays=5
    BadFlags=C:\Data\MyFlags.csv
wam_match_l2  Points.csv  C:\CAL\2009\MERIS2009\MER*.hdf algal_1  algal_2  maxDiffDays=5
```

These examples use the same points file *Points.csv*, specify the time limit of 5 days and use the standard *chlor_a* variable for SeaWiFS and MODIS-Aqua but *algal_1* and *algal_2* for MERIS. The MERIS files have been previously transformed with the following command:

```
wam_convert_n1   MER_RR__2*.N1 algal_1  algal_2  l2_flags
```

Sample commands that also select and record reflectances that can be used to evaluate other algorithms:

```
wam_match_l2 Points.csv E:\Sat\2007\S2007\S*.hdf maxDiffDays=5 Rrs_412 Rrs_443 Rrs_490
Rrs_510 Rrs_555 Rrs_670 sensor=SEAWIFS

wam_match_l2 Points.csv E:\Sat\2007\A2007\A*.hdf maxDiffDays=5 Rrs_412 Rrs_443 Rrs_488
Rrs_531 Rrs_547 Rrs_667 sensor=MODISA

wam_match_l2 Points.csv E:\Sat\2007\MERIS\RR\MER*.hdf reflec_1 reflec_2 reflec_3 reflec_4
reflec_5 reflec_6 reflec_7 reflec_8 reflec_9 reflec_10 reflec_12 reflec_13 reflec_14 algal_1
algal_2 maxDiffDays=5 sensor=MERIS
```

wam_match_multiband

```
wam_match_multiband vers. 3.4
Usage: wam_match_multiband PointFile ListFile
```

```
 where PointFile is a CSV file with Lon,Lat,Date,Time,...
 ListFile has a list of paths and matching patterns for image files, e.g.
   C:\Sat\MODISA\Rrs412\A2012*9km
   C:\Sat\MODISA\Rrs443\A2012*9km
   C:\Sat\MODISA\Rrs488\A2012*9km
   C:\Sat\MODISA\Rrs547\A2012*9km
Option minNValid=X changes the default minimum number of valid pixels to X from
the default value of 3. minNValid must be between 1 and 9.
Option maxDiffDays=D changes the maximum match-up time difference to D days from
 the default value of 30.
Example:
 wam_match_multiband Points.csv ListFile.txt maxDiffDays=5
```

wam_match_multiband is a command line utility that is related to *wam_match_nearest* and *wam_match_l2*. Like *wam_match_nearest* it finds the statistics for the nearest valid match-up in time and space, however, instead of using one type of image files, it does it for multiple data files, such as a set of remote sensing reflectance *Rrs* bands (*Rrs443*, *Rrs490*, *Rrs510*).

As input it needs 2 text files. The first is a typical point file (a text file or CSV file) with station information in the following order: Longitude, Latitude, Date, Time followed by any number or additional fields separated by the same separator, e.g. comma. Note that the first line is assumed to be a header and is ignored. A sample *Point* file is given below. You can exclude lines from being used by using the # character in the beginning of a line.

```
Lon,Lat,Date,Time,Station,Chl
-118.159,33.495,10/26/2008,21:45,1,0.368
#-118.105,33.434,10/26/2008,22:45,2,0.436
#-118.011,33.402,10/26/2008,23:45,3,0.623
-114.406,31.325,10/27/2008,15:45,4,2.408
```

The 2nd input is another text file that specifies the matching pattern of files. It has no header and the matching pattern. You can use the matching pattern to limit the number of files to be tested. The directory may be the same for all the required bands. In the latter case you must be careful to match the right band with the matching pattern. A sample list file looks like that:

```
E:\L3\MODISA\L3\Daily\Rrs_9\2010\A*412_9km
E:\L3\MODISA\L3\Daily\Rrs_9\2010\A*443_9km
E:\L3\MODISA\L3\Daily\Rrs_9\2010\A*488_9km
E:\L3\MODISA\L3\Daily\Rrs_9\2010\A*531_9km
E:\L3\MODISA\L3\Daily\Rrs_9\2010\A*547_9km
E:\L3\MODISA\L3\Daily\Rrs_9\2010\A*667_9km
```

This program tries to find the nearest satellite images with sufficient valid data to create statistics for the 3 x 3 pixel window centered at the station. This program is typically applied to daily global images and it sorts all images by their proximity to the date of the station. It then goes through the sorted list of images and tries to find valid (cloud-free) data for each station. If the nearest in time image is cloudy or has no valid data, it switches to the next image in time (first forward in time and then back in time from the nearest match). This process continues until the first cloud-free image is found or until maximum allowed time delay (30 days) is reached.

wam_match_nearest

```
wam_match_nearest vers. 2.24
Usage: wam_match_nearest PointFile Pattern
  where PointFile is a CSV text file with Longitude, Latitude, Date, Time, etc.
  Pattern is a matching pattern of HDF4 or netCDF filenames to be used
Example:
wam_match_nearest Points.csv C:\Sat\SeaWiFS\L3\Daily\S*_9km
Optional arguments:
Add=Yes means that all rows are kept even if no matchup is found; default is No
AnyYear=1 where 1 means that year is not considered when
  calculating time difference and can be any year, e.g. a multi-year composite;
  0 is default and means that year is considered
SDSNumber=N uses Nth dataset (SDS) in the file, default is 0
DX=X sets the area of interest in pixels, DX x DX, default is 3 x 3
MaxDiffDays=X sets the max allowed time difference in days; default is 30
MinNValid=X sets the min number of valid pixels required inside DX x DX area
Plotting options: (if any of these is set, there will be a plot that you must
 manually close)
plotNthColumn=picks Nth column of the PointFile to plot as in situ data
Log=No makes Linear plots and statistics. Default is to use log10
xMin=X assigns X as minimum plotting limit in linear or log10 units
xMax=X assigns X as maximum plotting limit in linear or log10 units
yMin=X assigns X as minimum plotting limit in linear or log10 units
yMax=X assigns X as maximum plotting limit in linear or log10 units
MinMaxLine=No does not plot Min-Max range. Default is yes.
SaveBrackets=Yes saves median bracket points of the match-ups.
NBracketsMax=X sets the maximum number of bracket points; default is 50
SaveEMF=Yes saves plot as EMF. Default is to save only PNG.
SaveStatist=Yes saves statistics as CSV. Default is not to save.
ValidMin=X sets valid minimum. Default is to get it automatically.
ValidMax=Y sets valid maximum. Default is to get it automatically.
Verbose=Yes prints additional info
Example:
wam_match_nearest Points.csv C:\L3\Daily\A*_9km xMin=-2 xMax=2 yMin=-2 yMax=2
```

wam_match_nearest is a command line utility that is related to the GUI application *wam_match*. Please see section on *wam_match* in this document for more information. This command is typically applied to **Level-3** (e.g. daily) global images. You can also use it for **Level-2** datasets buts there is a specialized version for those - *wam_match_l2*.
wam_match_nearest reads a CSV file with station information (including *Longitude*, *Latitude*, *Date*, *Time* - in that order!). It then goes through the sorted list of images and tries to find valid (cloud-free) data for each station. It loads the first dataset in the file and evaluates pixels in, e.g., 3x3 pixel window. If the nearest in time image is cloudy in that location, i.e. it has less than *MinNValid* pixels are valid of the 3x3 pixel window centered at the nearest pixel, it switches to the next image in time (either after or before the current image). This process continues until an image with sufficient valid pixels is found or until maximum allowed time delay (default is 30 days) is reached and the process of finding match-ups for this point is given up. In contrast to *wam_match*, you don't need a fixed list of filenames and you don't need a fixed time lag. Instead,

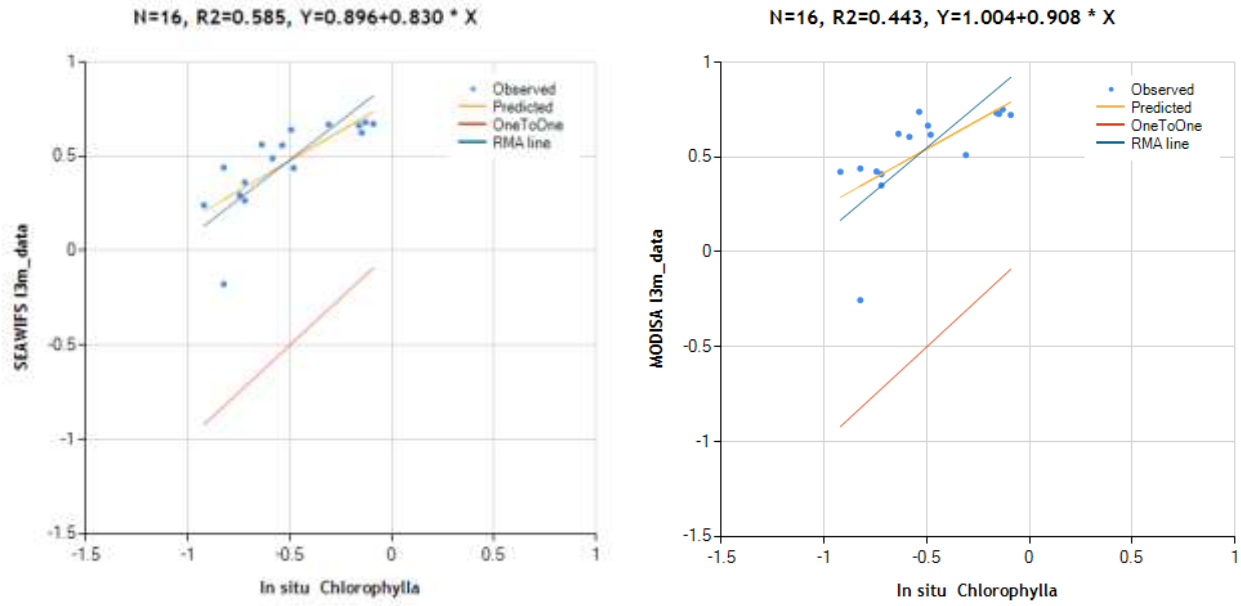
you can give a path to the matching image files, e.g. *C:\sat\SeaWiFS\L3\Daily\S1999*9km* and *wam_match_nearest* will find the nearest image in time with valid data corresponding to each station.

The matching HDF4 files may have more than one dataset (SDS) and in that case match-ups are provided for all of them. Datasets with a name matching "flags" will be skipped. It is assumed that the multiple datasets in a file have the same size and projection.

As an option, you can also make a plot of the match-ups at the same time when finding them. Plots have very many formatting options and are better to make AFTER you find the match-ups using some other software. However, if you know the plotting options, you can make decent plots right in *wam_match_nearest*. Here are 2 examples of finding match-ups and making plot using SeaWiFS or MODISA daily global images with respective outputs:

wam_match_nearest Points.csv SeaWIFS\L3\Daily\CHL_9\2005\S*.* maxdiffdays=5 plotNth=5 xMin=-1.5 yMin=-1.5 xMax=1 yMax=1

wam_match_nearest Points.csv MODISA\L3\Daily\CHL_4\2005\A*.* maxdiffdays=5 plotNth=5 xMin=-1.5 yMin=-1.5 xMax=1 yMax=1



Option *maxdiffdays=5* specifies to use maximum time difference of 5 days for the match-ups, *plot=yes* makes plots, *plotNth=5* specifies to use data from column 5 of the point file for the in situ variable, *xMin=-1.5* specifies to use horizontal plotting range minimum of -1.5, *yMin=-1.5* specifies to use vertical plotting range minimum of -1.5, *xMax=1* specifies to use horizontal plotting range maximum of 1, *yMax=1* specifies to use vertical plotting range minimum of 1. Please note that the default is to plot in log10 scaling. For plotting in linear scaling use *log=no*. Please note the change in the plotting range values when switching from log10 to linear plotting. As you input Point file can have many columns with different variables, you need to specify the column number to be used in plotting (e.g. *plotNth=5*). The Mean value of the valid pixels in the 3x3 pixel window is plotted. Other statistics are in the output file. Note the axis labeling: "In situ" is used for the horizontal axis followed by the header string of the *plotNth* column (e.g. *Chlorophylla*). The vertical axis is labeled by the name of the sensor (if recognized, e.g. SEAWIFS or MODISA) followed by the image name. The Level-3 mapped images are often

named as *l3m_data* and therefore do not have the name of the actual variable, e.g. *chlor_a*. The example plots above show that for the data in this *Points.csv* file the standard satellite-derived *chlor_a* estimates are significantly over-estimating the *in situ* values. The blue dots correspond to the *in situ* value versus the mean of the valid pixels, the red line is the one-to-one line, the yellow line is the ordinary linear regression and the blue line is the reduced major axis linear regression. You need to close the plot window (by clicking on the x sign in the upper right corner). You will be given warnings if any of the points are outside of *xMin*, *xMax*, *yMin* or *yMax*, respectively. For plotting purposes the outside value is capped to the respective limit. For example, negative values would prevent plotting in log10 scale, however, if *xMin* in log10 scale has been set to -2 (i.e. 0.01 in linear scale) then those points below *xMin* will be shown on the left edge of the window and a warning will be printed on the command window.

For visualization of your matchups you can use *wam_read_match* and *wam_match* (with the *Load from CSV* option).

wam_pixelwise_match

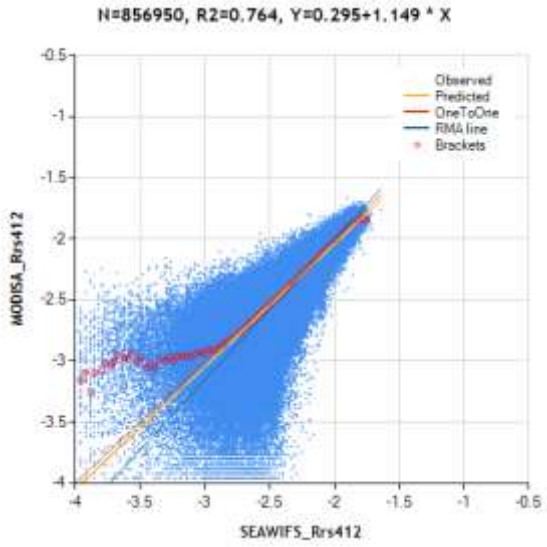
```
wam_pixelwise_match, vers. 1.18
Usage: wam_pixelwise_match Pattern1 Pattern2 [Mask.hdf]
  Pattern1 and Pattern2 are the matching filename patterns
  Mask.hdf is a HDF file that shows area to be used with pixel value = 1
Options:
Sds1=N reads the Nth dataset from files of series 1, default is 0
Sds2=M reads the Mth dataset from files of series 2, default is 0
Ratio= assigns a limit to val1/val2 to be included, e.g. Ratio=10 means that
  points of ratio val1/val2 outside 10 and 0.1 are excluded
ShowRatio=Yes shows the ratio Rrs/Rrs versus Rrs/Rrs instead of Rrs versus Rrs
LimitXFrom=L eliminates all matching points with X < L
Band1DenomName=Rrs_XXX sets a denominator band to Variable1 for band ratio
Band2DenomName=Rrs_XXX sets a denominator band to Variable2 for band ratio
BandRatioLessThan=Y sets upper limit to band ratio, e.g. Rrs_488/Rrs_547, e.g.
  default BandRatioLessThan=1.3 corresponds to Chl >~ 1 according to OC3M
Default is not to exclude any points
xMin=X assigns X minimum plotting limit in log10 units
xMax=Y assigns X maximum plotting limit in log10 units
yMin=X assigns Y minimum plotting limit in log10 units
yMax=Y assigns Y maximum plotting limit in log10 units
SaveBrackets=Yes saves the bracket points as CSV
NBracketsMax=X sets the maximum number of bracket points; default is 50
SaveEMF=Yes saves the plot in EMF in addition to PNG
SaveStatist=Yes saves statistics as CSV
From and To limit the seasonal range to be used.
The units can be either Months (with monthly data) or Julian days
(if higher frequency data are used), e.g.
From=1 To=31 will use days 1-31, i.e. January, if using data with
higher than monthly frequency;
From=3 To=9 will only use months 6 to 9 if using monthly data;
From=10 To=2 will only use months 10, 11, 12, 1, 2, i.e. the winter period
  if using monthly data.
Lag=X will lag 1st series by X steps relative to series 2,
```

Lin=yes uses Linear/Linear plot and statistics instead of Log/Log
MeanBrackets=yes uses Mean instead of Median to calculate the bracket values
Median1=Yes applies 3x3 median filter to series 1 images
Median2=Yes applies median filter to series 2 images
Matching pairs of points are saved in 'match_2sets_*.csv' in the current folder
Median bracket points are saved in 'median_brackets_*.csv' in the current folder

Examples:
wam_pixelwise_match data1*.hdf data2*.hdf Mask.hdf
wam_pixelwise_match E:\A*Rrs_443* Mask.hdf xMin=-4 xMax=-1 yMin=-4 yMax=-1

wam_pixelwise_match finds pixelwise matches between 2 sets of images and optionally plots the scatter plot. For example, if you want to compare SeaWiFS and MODIS-Aqua *Rrs_412* values in daily mapped global images in your area specified by a mask in *Mask.hdf*, you can issue a command like:
wam_pixelwise_match SeaWiFS\L3\Daily\Rrs_9\2004\S*Rrs_412* E:\L3\MODISA\L3\Daily\Rrs_9\2004\A*Rrs_412* Mask.hdf varname1=Rrs412 varname2=Rrs412 xMin=-4 xMax=-1 yMin=-4 yMax=-1

Here varname1 and varname2 are just used as the plotting labels and xMin, xMax, yMin, yMax are the plotting limits in log-scale. The output is 2 CSV files (the actual match-ups and their median-median brackets) and the plot as a PNG and EMF file. Be careful not to pick too many matching points as you may have millions of matching points and then you may run out of memory and fail to create the EMF file. A sample PNG plot is below. It shows that the SeaWiFS and MODISA values tend to be similar, i.e. the central tendency is close to the one-to-one line but that there is a lot of scatter, especially at low values.



wam_pixelwise_match_add
wam_pixelwise_match_add, vers. 1.5
Usage: wam_pixelwise_match_add Dir1 Dir2 Matchfile
Dir1 and Dir2 are the directories of matching filenames
Matchfile is a previously saved CSV file with Lon,Lat,Date,Time,x,y,ImageName1 ,Val1,ImagName2,Val2...

Options: Sds1=SDS1 Sds2=SDS2,
 Sds1=N reads the Nth dataset from files of series 1, default is 0
 Sds2=M reads the Mth dataset from files of series 2, default is 0
 Varname1= assigns series1 variable name, default is to use image name
 Varname2= assigns series2 variable name, default is to use image name
 xMin=X assigns X minimum plotting limit in log10 units
 xMax=Y assigns X maximum plotting limit in log10 units
 yMin=X assigns Y minimum plotting limit in log10 units
 yMax=Y assigns Y maximum plotting limit in log10 units

wam_pixelwise_match_add is a follow-up to *wam_pixelwise_match* as it takes the CSV output file with the matching pixels, produced in *wam_pixelwise_match*, uses the day, X and Y coordinates of the matching pixels from that file and finds corresponding match-ups of other bands. For example, in the example for *wam_pixelwise_match* we found pixelwise matches between SeaWiFS and Aqua using *Rrs_412* bands. We now want to add other bands to the file and we can that step-wise, first adding *Rrs_443*, then the next bands and so on.

wam_read_match

wam_read_match vers. 2.41
Usage: wam_read_match MatchFile [Options]
 where MatchFile is output from matching, i.e. a CSV text file with Lon,Lat,..
Optional arguments:
All=Yes skips filtering and picks ALL points. Default is no.
Default is to pick only 'good' points, i.e. SatMean>0, inSitu>0
BadFlags=Flag1,Flag2,... Makes pixels with those flags invalid, e.g.,
 the following eliminates pixels not used by NASA for standard L3 files:
BadFlags=ATMFAIL,LAND,HIGLINT,HILT,HISATZEN,STRAYLIGHT,CLDICE,COCCOLITH,HISOLZEN
,LOWLW,CHLFAIL,NAVWARN,MAXAERITER,ATMWARN,NAVFAIL,FILTER
MaxDiffDays=X keeps only those with <= time diff. Default is no screening.
NValidFrom=X sets the min number of valid pixels required, default = 1
NValidTo=Y sets the max number of valid pixels allowed, default = 9
NCleanFrom=X sets the min number of unflagged pixels required, default = 0
NCleanTo=Y sets the max number of unflagged pixels allowed, default = 9
plotNthColumn=picks Nth column of the MatchFile to plot as IN SITU column
satVariableToPlot=VariableName picks VariableName as the SATELLITE column
insituVarAtLeast=X selects matchups with in situ variable at least X
insituVarLessThan=Y selects matchups with in situ variable less than Y
satVarAtLeast=X selects matchups with satellite mean at least X
satVarLessThan=Y selects matchups with satellite mean less than Y
satRelRangeAtLeast=Y selects only with (SatMax-SatMin)/SatMin >= Y
satRelRangeLessThan=Y selects only with (SatMax-SatMin)/SatMin < Y; Default is Y=1.0
ratioSatToInsituAbove=X selects matchups with ratio SatMean/Insitu > X
ratioSatToInsituBelow=Y selects matchups with ratio SatMean/Insitu < Y
SaveAll=Yes saves the input and the output of Algorithms. Default is not to save.
SaveGood=Yes saves the good (positive) matchups. Default is not to save.
SaveSelected=Yes saves the selected matchups. Default is not to save.
SaveBrackets=Yes saves median bracket points of the selected.
NBracketsMax=X sets the maximum number of bracket points; default is 50

For Plotting:
xMin=X assigns minimum plotting limit in log10 or linear units
xMax=X assigns maximum plotting limit in log10 or linear units
yMin=X assigns minimum plotting limit in log10 or linear units
yMax=X assigns maximum plotting limit in log10 or linear units
Log=No makes Linear plots and statistics. Default is to use Log10
MinMaxLine=No makes no vertical line for Min-Max. Default is yes.
SaveEMF=Yes saves Chart as EMF. Default is not to save EMF.
SavePNG=No does not save Chart as PNG. Default is to save PNG.
SaveStatist=Yes saves Statist.txt. Default is not to save Statist.txt.
Sensor type is normally guessed from the satellite image file name but can be set
with option Sensor=SSS where SSS can be SEAWIFS, OCTS, MERIS, MERISL2,
MODIST, MODISA, CZCS, OSMI, VIIRS, GLI, MERISNASARR, MERISNASAFRS
You can apply various algorithms to Rrs data. Default is none.
Algorithm=AAA specifies algorithm to be applied to Rrs data.
Options are: OC2, OC3, OC4, ChlCalFit3, ChlCalFit4, ChlNIRR2, CHLRGCI,
ChlSpgAnt3, ChlSpgAnt3Blended, ChlSpgAnt4, ChlSpgAnt4Blended, QAA
ChlFromAph440SPGSO, NppVGPM, NppSpgAnt, NppLee
For NPP models you can specify the zEuModel=X that specifies the following models
for calculating zEu from Chl: 0=Morel&Berthon, 1989; 1=Morel&Maritorena, 2001;
2=SPG_SO_2004; 3=SPG_SO_2006; 4=Fixed_50m; 5=N/A; 6=SPG_SO_2012
Input arguments for NppVGPM are guessed from column header names
but can be set as nppName=A chlName=B parName=C sstName=D
Examples:
wam_read_match match_VIIRS.csv xMin=-2 x Max=2 yMin=-2 yMax=2 maxDiffDays=5
nValidFrom=3
... is plotting default chlor_a with limits in Log10 units
wam_read_match match.csv xMin=1 xMax=4 yMin=1 yMax=4 algorithm=NppVGPM
chlName=chlNPGANT4
... is applying VGPM using column=chlNPGANT4 as input for Chl

wam_read_match reads the output from *wam_match_l2* or *wam_match_nearest*, makes plots and calculates statistics for a selected pair of variables. Typically it is used to plot satellite-derived Chla against *in situ* Chla. However, it has very many options and can be used to calculate new variables (e.g. Chla from Rrs) or NPP from various inputs, create and saves subsets, etc. The format of match-up dataset is a CSV (comma separated file) with the last columns reserved for the L2-flags. You can use various columns to calculate new variables, e.g. Chl from Rrs using a different algorithm and the new variable will be added as a new column after the last column. If you want to use that new file again as input to wam_read_match, you need to load the file into Excel, cut and move the new column to the left of the L2-flags columns, and save again as CSV.

Multiple options allow selecting various subsets of the match-up points. When using the option All=Yes, no screening is done and all points are read but plotting is still limited to positive satellite mean and *in situ* data pairs (as *log10* cannot be taken from negative numbers).

The subsets of "selected" match-ups can be found using many criteria and their combinations.

These "selection" parameters that can be changed in the command line are the following:

- maxDiffDays=D sets the maximum time delay to D days;
- nValidFrom=M sets the minimum number of valid pixels required within the 3 x 3 pixel window;
- nValidTo=M sets the maximum number of valid pixels required within the 3 x 3 pixel window;
- nCleanFrom=M sets the minimum number of totally unflagged pixels required within the 3 x 3 pixel window;
- nCleanTo=M sets the maximum number of totally unflagged pixels required within the 3 x 3 pixel window;

The following are the "plotting" parameters that can be set:

- xMin=X assigns minimum plotting limit in log10 or linear units;
- xMax=X assigns maximum plotting limit in log10 or linear units;
- yMin=X assigns minimum plotting limit in log10 or linear units;
- yMax=X assigns maximum plotting limit in log10 or linear units;
- Log=No makes Linear plots and statistics. Default is to use Log10;
- MinMaxLine=No makes no vertical line for Min-Max. Default is yes.
- SaveEMF=Yes saves Chart as EMF. Default is not to save EMF.
- SavePNG=No does not save Chart as PNG. Default is to save PNG.

The following are the "saving" parameters that can be set:

- SaveStatist=Yes saves Statist.txt. Default is not to save Statist.txt;
- SaveGood=Yes saves the good (positive) matchups. Default is not to save.
- SaveSelected=Yes saves the selected matchups. Default is not to save.
- SaveBrackets=Yes saves median bracket points of the selected.

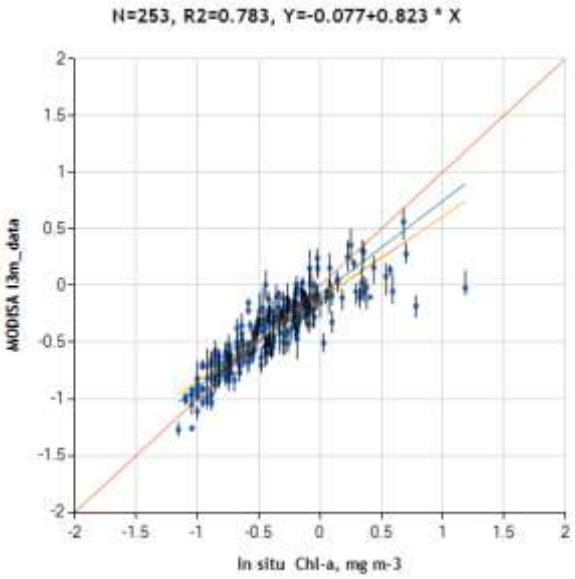
The following are the useful to find outliers, save to a separate dataset and eliminate from the main dataset by preceding the line with the # character:

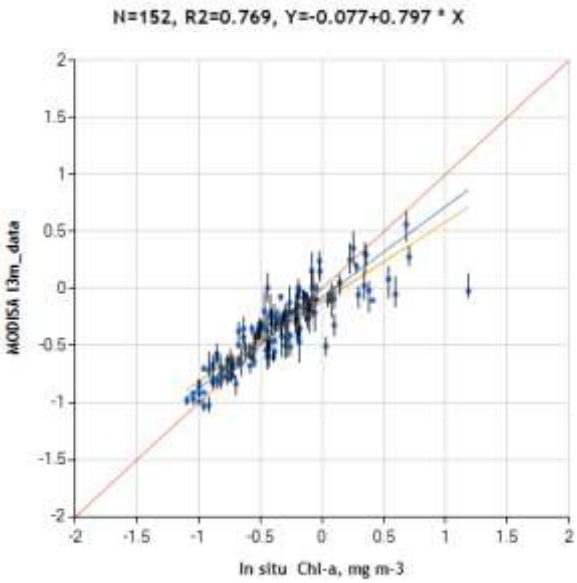
- insituVarAtLeast=X selects matchups with in situ variable at least X.
- insituVarLessThan=Y selects matchups with in situ variable less than Y.
- ratioSatToInsituAbove=X selects matchups with ratio SatMean/Insitu > X.
- ratioSatToInsituBelow=Y selects matchups with ratio SatMean/Insitu < Y.
- SaveSelected=Yes saves the selected matchups. Default is not to save.

Example:

```
wam_read_match MODISA_l3m.csv  xmin=-2 xmax=2 ymin=-2 ymax=2 maxdiffdays=1
```

The command above reads match points from a file and finds 253 of them as "good" after global screening and then selects 152 points (out of 253) in secondary screening by including only those points with 1 day or less time difference between the in situ measurement and the satellite. As satellite data were daily level 3 data, timing to the nearest day is used. Statistics is calculated for the subset of 253 points (with 5 day maximum difference) and to the screened dataset with maximum 1day difference (152 points). It appears (you can check the statistics in the plot headers) that the additional screening of matchups from maximum 5 days to maximum 1 day in time difference did not improve the correlation. The correlation actually became slightly worse (R2 from 0.783 to 0.769) and the slope also slightly worse (away from the ideal slope=1). Those differences are probably not significant. We can conclude that when using global 9 km imagery, the measured chlorophyll distribution does not change significantly between 1 and 5 days, therefore, we can use 5-day match-ups that provide more match-ups in the statistics.





Example:

```
wam_read_match MODISA.csv xmin=-2 xmax=2.5 ymin=-2.5 ymax=2 maxdiffdays=1
ratioSatToInsituBelow=0.1 saveselected=yes
```

The above command above reads match points from a file and selects and plots only those that are less than 0.1 of the in situ value, i.e. underestimated by 10 times or more. These will be saved in a new file and these can be examined to find the reason of this gross underestimation.
Note: the *l2_flags* must be the last "variable" in the file. You can even "add" columns to the match file by matching with level-3 files using *wam_match_nearest* with the match file but in order to use *wam_read_match* on the resulting file you need move l2_flags to the last column, e.g. by to loading the file into Excel, moving the l2_flags to the end and saving the file as CSV.

wam_pixel_extract

```
wam_pixel_extract  vers. 1.0
Usage: wam_pixel_extract Pattern  X   Y
Pattern is a matching pattern of HDF4 or netCDF files
X  is integer x-coordinate (0 at left)
Y  is integer y-coordinate (0 at top)
```

wam_pixel_extract extracts a single pixel value specified by its X and Y coordinates x is zero at left and y is zero at top) for a series of images specified by a matching set of HDF4 or netCDF filenames. For examples:

```
wam_pixel_extract  Images\S*.hdf   10  20
```

extracts pixel values of (10, 20) in all matching images and saves in the current directory as a CSV text file.

wam_statist_grid

```
wam_statist_grid  vers. 3.0
```

```
Usage: wam_statist_grid FilePattern
Options:
DeltaLon=X ; default is 10
DeltaLat=Y ; default is 2
LonUL=X LatUL=Y LonLR=X LatLR=Y specify upper left and lower right corners;
  default is to use the whole image.
Mask=mask specifies HDF4 file with pixel value=1 as area of interest
Example: wam_statist_grid *.hdf DeltaLon=360 DeltaLat=60
The example uses Lat bands of 60 deg around globe (note DeltaLon=360)
Note that Lon is before Lat !
Mask is a HDF file that specifies region of interest with pixel value 1
You can specify which dataset to read with sds=X; default is 0
```

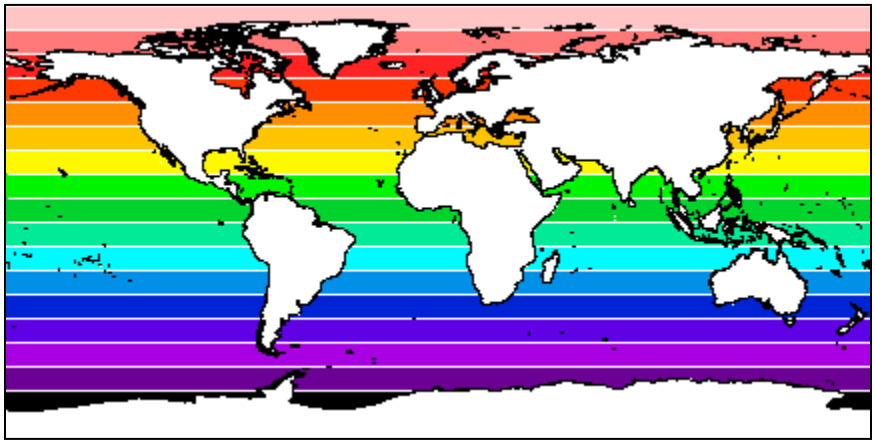
wam_statist_grid is one of several image statistics programs in WIM. In the GUI program *wam_statist* the areas of interest are specified with the mask image where each individual area of interest is specified by its pixel value (e.g. 1, 2, 3,…). If you want to calculate statistics, say, for each 10 degree by 10 degree square then you would need to manually create a masks image with each of the 10 degree x 10 degree squares specified by a respective pixel value (1, 2, 3, …). Clearly, it becomes tedious manual work. In contrast, *wam_statist_grid* does not need a list file as it uses a filename pattern to select files and it does not need a mask file as it assumes rectangular areas of interest specified by their longitudes and latitudes. For example, a *FilePattern* of *S*.hdf* or *C:\Sat\SeaWiFS\L3\Monthly\CHLO_9\S*.hdf* would use all matching SeaWiFS files. The default is to use *DeltaLon* of 10 degrees and *DeltaLat* of 2 degrees and cover the whole image. The latitude and longitude range of a smaller grid can be specified in the command line. For example, you may use global images but be interested in image statistics in a certain region, e.g. latitude range of 24-28 N and longitude range of 84-90 W with a 1 x 1 degree grid.

To run it, type:

wam_statist_grid S*.hdf DeltaLon=1 DeltaLat= 1 LonUL=-90 LatUL=28 LonLR=-84 LatLR=24

Note that [LonUL, LatUL] specify the upper-left corner and [LonLR, LatLR] specify the lower-right corner and that longitude values are before the latitude values.

Another example: we want to calculate statistics for 10-degree latitude bands for the whole globe like shown below.



We will use the following command:

wam_statist_grid S*.hdf 360 10

Note that we use 360 degrees for `DeltaLon` and 10 degrees for `DeltaLat` and use the whole image .

The program tries to guess the range of valid values and it should work for common images types. The valid min and max values are reported for the first image. If they are not correct then the source code needs to be updated and the program recompiled.

Output is written to a file *statistics_grid.csv* in comma separated (CSV) format and should be easy to import into a spreadsheet program like MS Excel. The output name inclides the SDS number (e.g. 00 for default SDS=0). The output has the following columns:

Image,SYear,EYear,SDay,EDay,Lat,Lon,Nin,Nout,Min,Max,Mean,StDev,Median

Here *SYear* and *EYear* specify, respectively, the start and end year of the image, *SDay* and *EDay*, the start and end day of the image, *Lat* and *Lon* are, respectively, the center coordinates of the grid rectangle, *Nin* and *Nout* are the number of pixels within the valid range and outside the valid range, respectively. Note that the output is sorted first by image, then by latitude and then by longitude. The Lat and Lon specify the center of each grid cell.

Please note that the output file is sorted by Image and in order to plot time series for each grid cell you need to sort by Latitude and by Longitude. A separate utility is provided for that with the following syntax:

sortGrid Unsorted_grid_time_series [ColumnNo]

where *ColumnNo* is the column number to pick. Default is to use *ColumnNo* 13 that means Median. Other options for *ColumnNo* are: 7 for *Nin*, 8 for *Nout*, 9 for *Min*, 10 for *Max*, 11 for Mean, 12 for StDev, 13 for Median. The output filename is using the same pattern with "sorted.csv" added to it.

The output format has all the longitude cells for a particular image (time) and Latitude value in the same row.

wam_statist_mask

wam_statist_mask vers. 1.9

```
Usage: wam_statist_mask FilePattern
Options:
Mask=mask specifies HDF4 file with pixel value=1 as area of interest
Ice=Pattern specifies the pattern of ice filenames
sds=X specifies dataset # to read; default is 0
validMin=X overrides valid min value
validMax=Y overrides valid max value
sds=All to read All datasets in the file
 All option currently works only with HDF4
Total=yes adds the sums of PixelValue * PixelArea,
 total areas of valid (In)and invalid (Out) pixels
```

wam_statist_mask is a simplified command line version of the GUI program *wam_statist*. Command line programs are easier to automate and use in batch files. It will calculate statistics for a series of matching HDF4 or netCDF files specified by the *FilePattern* argument. In contrast to *wam_statist* it only allows to use one area of interest that is specified by pixel value 1 in the mask image. Alternatively, when mask is not specified, it calculates statistics for the whole image.

wam_statist_sta

```
Usage: wam_statist_sta StationsList Pattern [OutputFile]
  StationsList is a list of stations with Lon, Lat[, Date, Time,
StationName]
  Pattern is a matching pattern of HDF file names
  OutputFile is text file where output is saved
  If OutputFile is not given then output is to the screen
```

wam_statist_sta is a utility for quickly calculating statistics for a series of points (stations) and a series of images. It uses a list of points (stations) given by their longitude, latitude and name, and a set of matching images, to calculate statistics for 3 x 3 pixel areas centered at the stations for all the images. The statistics and all the 9 pixel values are saved in a CSV (comma separated value as text) file suitable for importing into a spreadsheet (e.g. Excel). Please note that a related GUI program *wam_statist* includes most of the functionality of *wam_statist sta* but does not save the individual pixel values.

You need to create a station list file with coordinates and have a series of images. A sample station list file is a CSV (comma separated values) file and it looks like this:

```
Lon,Lat,Sta
-113.667,30.333,GOC
```

Note that longitude is before latitude! You can have multiple stations specified in a station list file, e.g.

Longitude,Latitude,Station

-117.3050,32.9567,Station_1

-117.3950,32.9133,Station_2

You can create a text file with a single imaginary station *MyStation.txt* (e.g. like the station GOC in the example above) and run *wam_statist_sta* with the SeaWiFS CHLO_9 files provided on the WAM DVD:

wam_statist_sta MyStation.txt C:\Sat\SeaWiFS\L3\Monthly\CHLO_9\S1998*.hdf Out.csv

where *MyStation.txt* is a text file with longitudes, latitudes and station names (in that order), *C:\Sat\SeaWiFS\L3\Monthly\CHLO_9\S1998*.hdf* is a pattern of matching filenames to be used, *Out.csv* is the name of the output file. Output is a text file (CSV) and that can be loaded into Excel or viewed in a text editor (e.g. Notepad).

Note that in other localizations ("Cultures") comma is used instead of the decimal point and you must use tab instead of comma as the separator between values. The output file is also in the CSV format and has the following header:

Image,SYear,EYear,SDay,EDay,Station,Nin,Nout,Min,Max,Mean,StDev,Median,Pointvalue,
Pixel_1,Pixel_2,Pixel_3,Pixel_4,Pixel_5,Pixel_6,Pixel_7,Pixel_8,Pixel_9

Here *SYear* and *EYear* specify, respectively, the start and end year of the image, *SDay* and *EDay*, the start and end day of the image, *Nin* and *Nout* are the number of pixels within the valid range and outside the valid range, respectively. *Pointvalue* is the pixel value in the nearest pixel that is the center of the 3 x 3 pixel area, and the last 9 columns are the individual pixel values.

Note that the output is sorted first by image and then by station. If you want to sort first by station, for easy plotting of time series for each station, you can use another WAM utility *sortstas* or sort the file in Excel.

wam_make_mask

```
wam_make_mask vers. 1.0
Usage: wam_make_mask xSize ySize lonFrom lonTo latFrom latTo
[maskFile]
```

Labels the mask image used in *wam_statist*: writes the sequence number into the middle of each mask, saves as HDF and PNG. It first counts all pixels between 1 and 254 (excluding 255), calculates mean X and Y coordinates for them. It then converts pixel value 0 to 255 (black to white), creates coastlines using the moderate resolution coastlines and overlays coastlines in black. It then annotates the image with the mask number using the mean X and mean Y. It is supposed to be in the middle of each cluster (mask) of pixels.

wam_label_mask

```
wam_label_mask vers. 1.1
Usage: wam_label_mask MaskFile
```

Labels the mask image used in *wam_statist*: writes the sequence number into the middle of each mask, saves as HDF and PNG. It first counts all pixels between 1 and 254 (excluding 255), calculates mean X and Y coordinates for them. It then converts pixel value 0 to 255 (black to white), creates coastlines using the moderate resolution coastlines and overlays coastlines in black. It then annotates the image with the mask number using the mean X and mean Y. It is supposed to be in the middle of each cluster (mask) of pixels.

sortmasks

```
Usage: sortmasks unsorted_file
```

sortstas

```
Usage: sortstas unsorted_file
```

sortgrid

```
Usage: sortGrid Unsorted_grid_time_series [ColumnNo]
Default is to use ColumnNo=13 that means Median
ColumnNo values: 7 for Nin, 8 for Nout, 9 for Min, 10 for Max, 11
for Mean, 12 for StDev, 13 for Median
```

The output format from the WAM statistics programs, e.g. *wam_statist*, is effective to produce but cumbersome to use with simple plotting routines, e.g. in MS Excel. To sort the output file into a format more suitable for plotting in Excel, use a WAM utility *sortmasks*. The sorted file is better for plotting in a spreadsheet like MS Excel.

sortstas and sortgrid do similar sorting of the output from wam_statist_sta and wam_statist_grid, respectively.

To compile *sortstas*, don't use *c.bat* as MWIM.dll is not used here. The command for building *sortstas* is:

csc /out:%WIMSOFT%\sortstas.exe sortstas.cs

3.2.10 Screening Level-3 data according to quality flag

wam_screen_goes1112

```
wam_convert_goes1112 vers. 1.0
Usage: wam_screen_goes1112 Pattern [Threshold]
  Default threshold is 252, corresponding to cloud % of 0.01
Examples:
  Cloud percentage: 0.01%  0.1%   1.0%   2.0%  5%  10% 20% 50%
  Thresholds:       252    237    181    157   122  95  67  30
```

Uses output from *wam_convert_goes1112* and screens SST data according to cloud probability threshold. Keeps only SST values with equal and higher corresponding threshold value, i.e. less or equal cloud probability. See ftp://podaac.jpl.nasa.gov/sea_surface_temperature/goes/ for GOES 11-12 data and documentation.

wam_screen_mask

```
wam_screen_mask vers. 1.12
Usage: wam_screen_mask Pattern Mask.hdf
  where Pattern is a matching pattern of HDF4 or netCDF files to
use
  and Mask.hdf is a mask file in HDF4
Valid area is masked with pixel value 1
Option Delete=No saves the original files from deleting
Option Move=No saves the original files from moving
```

wam_screen_mask is a utility for working with standard ocean color Level-2 satellite data. It is common that many Level-2 files are either cloudy, are outside of our area of interest or have no valid data for various reasons (e.g. due to low light in polar areas). These files may take up lots of disk space but have no useful data. *wam_screen_mask* sorts files into separate folders after which files with no valid data can be deleted. The valid pixels are currently counted only for the "*chlor_a*" and "*sst*" data setd (SDS). The files with at least 1 valid pixel in the valid area specified by a mask image (with pixel value = 1) are moved to folder "*Good*", files with no matching dataset into folder "*Bad*", files that are completely outside the area of interest are moved to "*Outside*" and those with no valid data to "*NoValidData*". All folders other than "*Good*" can be deleted. After that you should run *wam_compress_hdf* on the files in the "*Good*" folder.

wam_screen_pf

wam_screen_pf vers. 3.2
Usage: wam_screen_pf Pattern [qualityThreshold]
Pattern is matching pattern of either HDF4 or netCDF filenames
Default qualityThreshold=4
quality_level: 2=worst 3=low, 4=acceptable, 5=best
V5.2 pathfinder_quality_level: highest is 7

Screens AVHRR Pathfinder 5 SST data files and saves in scaled byte images with Pathfinder SST scaling. Adds extension *_screened.hdf* to the output files. Before Pathfinder version 5.2 the SST and quality data were in separate files. These files need to be in the same folder. See PF5 user guide at http://www.nodc.noaa.gov/SatelliteData/pathfinder4km/userguide.html
Starting with version 5.2 the images are in the same file in datasets named *sea_surface_temperature*, *quality_level* and *pathfinder_quality_level*. Either one of the quality level images needs to be in the file but only the last one in the sequence of images will be used.

The default quality threshold is 4. The extra argument allows changing the quality threshold level. For the variable *quality_level* the meaning is the following: (0 = no SST data, 2=worst, 3=low, 4=acceptable, 5 = best quality data).

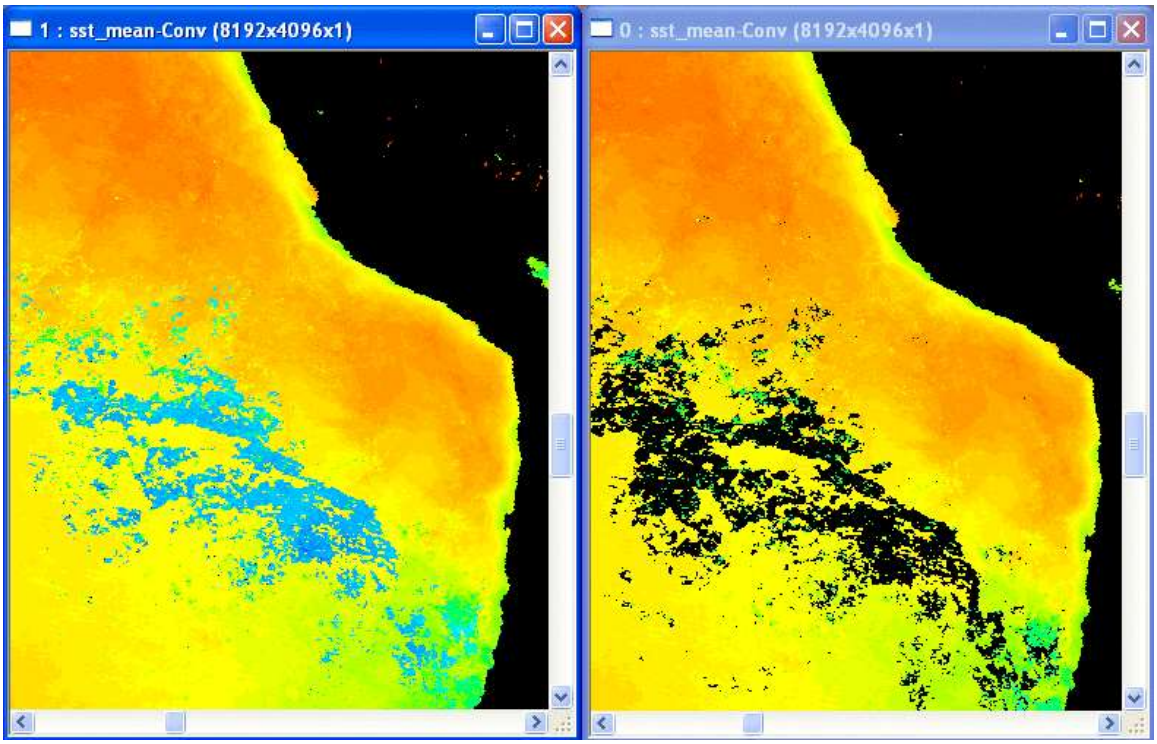
For the variable *pathfinder_quality_level* ranges from 0 to 7, where 0 is worst and 7 is best, and value -1 represents missing data.

wam_screen_sst_ocpg

wam_screen_sst_ocpg vers. 1.1
Usage: wam_screen_sst_ocpg Pattern
Sets pixel value to 0 if quality flag != 0

This command is for screening MODIS Terra and Aqua SST data processed by the Ocean Color Processing Group (OCPG). Pixels with the corresponding best quality flag (from 4 to 7 for Pathfinder, 0 for MODIS) will retain their value while pixels with lower quality flags will be nullified.
The *wam_screen_sst_ocpg* version screens the SST data in HDF files produced since 2006 by the NASA Ocean Color Processing Group. These files have the SST and the Flag datasets in the same file. Therefore, screening these is easier as there is no need to search for the other (quality flag) file.

The following example shows the unscreened MODIS sea-surface temperature (SST) image on the left and the screened SST image on the right. Both have been transformed to BYTE with the SST-Pathfinder color scaling. The unscreened image shows large areas with low SST off South America (light blue) which are probably due to cloud contamination. Clouds are almost always colder than the actual. On the screened image on the right the light-blue areas are mostly black, i.e. masked with no data. However, even on the screened image there are some green areas next to the masked areas which are probably the result of incomplete screening.*wam_screen_pf* does similar screening for the Version 5 AVHRR Pathfinder SST data. In contrast to the earlier Pathfinder SST data, the Version 5 data have pixels of various quality levels, many of which are obviously contaminated by clouds. The ascending data are daytime passes and should have better cloud detection. In order to have reliable (best quality) data we need to eliminate all SST pixels with *pathfinder_quality_level* value less than 7 or *quality_level* value less than 5. The screened SST data are saved in the Byte format with the *SST-Pathfinder* scaling.



3.2.11 Processing Level-1B MODIS data

wam_rgb_modis

```
Usage:
wam_rgb_modis Level-1B_File [rLow rHigh gLow gHigh bLow bHigh
yes/no_for_saving_3_comps target_proj_file]
The Level-1B_File is a filename pattern and all matching files
will be processed
The default is rLow=0, rHigh=255 gLow=0, gHigh=255, bLow=0,
bHigh=255
If 'yes/no_for_saving_3_comps' = 'yes', then the 3 individual
bands will be saved
If target_proj_file is given then the output images will be
mapped to the target projection.
```

wam_rgb_modis is a command-line program that creates one or a whole series of RGB (red-green-blue) composites from MODIS Terra or Aqua Level-1B calibrated radiances at 1 km (1KM), 500 m (HKM) and 250 m (QKM) resolutions. See separate document *Exercises_modis_250m.pdf* for details. It performs accurate co-registration of the different bands using an interpolation scheme that does scan by scan interpolation. The MODIS-Terra products used are MOD021KM, MOD021HKM and MOD02QKM and the respective MODIS-Aqua products have "MO" being replaced with "MY". *wam_rgb_modis* also handles the *crefl*-corrected radiance files (see a separate document *Exercises_modis_250m.pdf*).

The argument *Level-1B_File* is actually a matching pattern and if more than one files match the pattern, a series of files will be processed.

The problem when creating RGB images at the highest, 250 m resolution is that only the red band is measured at 250 m resolution while the blue and green bands are measured at 500 m resolution. Therefore, when creating RGB images at 250 m resolution both the QKM and HKM data are needed. The blue and green bands will be sharpened using the full-resolution red band. At 500 m and 1 km resolution the process of creating a RGB image is much easier as the lower resolution component bands are directly available from the respective HDF file.

The user can specify the *Low* and *High* values that will be used to stretch the R, G, B component, respectively. The default values are 0 and 255 for each. The *yes/no_for_saving_3_comps* option with a word 'yes' saves the 3 individual components in a separate file where they can be read with WIM and manipulated interactively. Please use 'no' or anything not starting with 'Y' or 'y' if specifying the next argument but not wanting to save the R, G, B components.

The last argument allows to remap the RGB image to a target projection. The target projection file must be a HDF file with a certain projection. With earlier L1B data the navigation had noticeable error without using an interpolated 250-m geo-location file with *zoom_modis_lat_lon* (see next section) program but with the more recent L1B data using *zoom_modis_lat_lon* does not seem to be necessary.

Please note that using atmospherically corrected radiances creates brighter and better-looking RGB images. It is described in *Exercises_modis_250m.pdf*.

zoom_modis_lat_lon

```
Usage: zoom_modis_lat_lon M?D03*.hdf
```

zoom_modis_lat_lon reads 1-km latitude and longitude arrays from the standard MOD03/MYD03 file and interpolates to 250-m resolution. The resulting file is saved as **.Lat_Lon.hdf*. The interpolated LLA file used to give better (more accurate) geo-referencing when using MODIS 250-m images. The application of *zoom_modis_lat_lon* was essential for accurate navigation with earlier L1B data but does not seem to be necessary with the more recent L1B data. See *Exercises_modis_250m.pdf* for details.

wam_turbidity

```
Usage: wam_turbidity Pattern [TargetProjectionFile] [slope]

The 'Pattern' is a filename pattern and all matching files will
be processed
If TargetProjectionFile is given then only a rectangle covering
the target area
will be processed,
    ... the result will be mapped to the target projection.
'slope' is for scaling of the resulting image. The default is
2.5.
At very high turbidity you may need to increase 'slope'.
The max valid value with slope of 2.5 is 254 x 2.5.
```

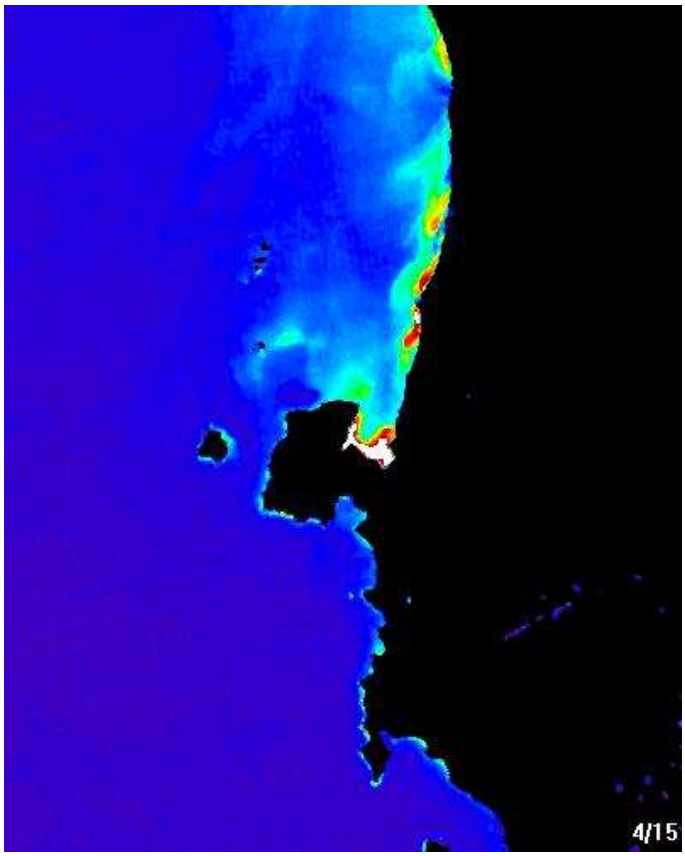
wam_turbidity is a command-line program for creating relative turbidity images from MODIS 250-m bands 1 and 2. See *Exercises_modis_250m.pdf* for details.

As input for *wam_turbidity* you need 3 files: HDF file with *crefl*-corrected MODIS bands 1 and 2, the corresponding MOD03/MYD03 file or the 250-m LLA file (generated with *zoom_modis_lat_lon*) with geo-referencing, a HDF file with the target projection.

If you have two matching geo-referencing files in the same folder (the original MOD03/MYD03 and the one generated with *zoom_modis_lat_lon*, then *wam_turbidity* will use the more accurate one. The program can process many sets of files from a single command. Therefore you can use a single filename or a pattern with wildcard characters to specify the files to be processed.

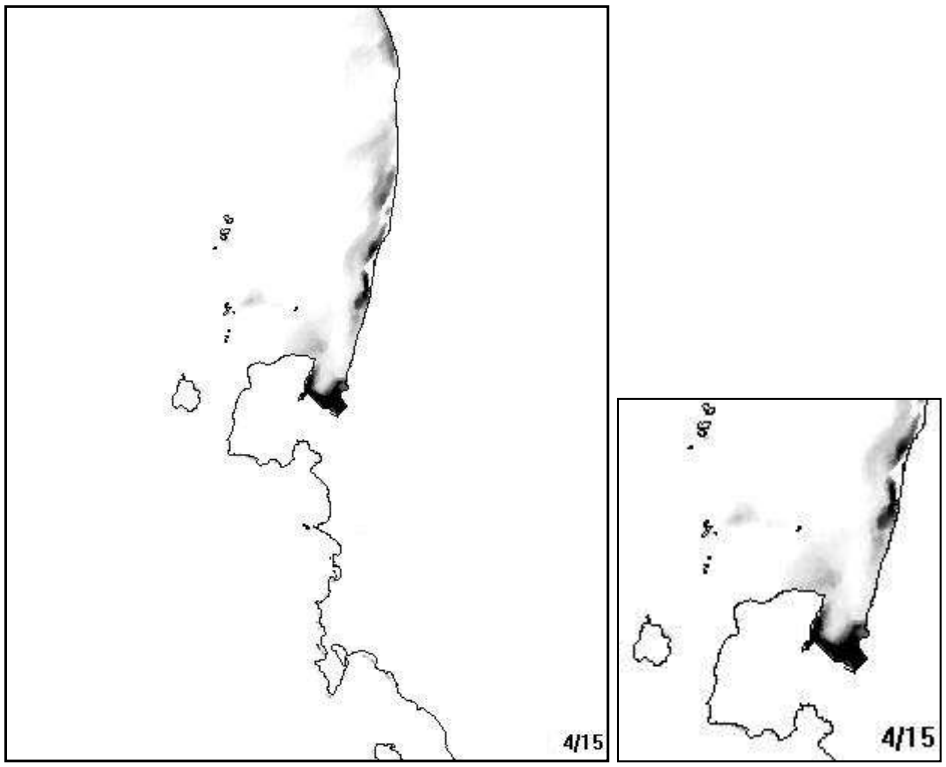
Please note that as input you need atmospherically corrected radiances produced with the *crefl* program. Please see *Exercises_modis_250m.pdf* for more information.

A sample output from *wam_turbidity* (showing relative turbidity in reflectance units) for area off Paracas in Peru looks like that:



You can see highest turbidity (white) inside the Paracas bay. This area of high turbidity was caused by a massive harmful algal bloom. Smaller patches of turbidity are visible along the coast, at least some of them caused by inflows from local rivers or canals.

The JPEG image is saved in grayscale: the highest concentrations are shown in black while low-concentration areas are shown as white, i.e. not highlighted at all. This is supposed to highlight the high turbidity areas, e.g. due to river plumes or harmful algal blooms.



The 2 figures above show the results from *wam_turbidity* using different target projections.

wam_turbidity_aster

```
Usage: wam_turbidity_aster pattern [target_projection_file] [slope]
The 'pattern' is a filename pattern and all matching files will be processed
If target_proj_file is given then only a rectangle covering the target area will
 be processed,
   ... the result will be mapped to the target projection.
'slope' is for scaling of the resulting image. The default is 1.
At very high turbidity you may need to increase 'slope'.
The max valid value with slope of 2.5 is 254 x 2.5.
```

wam_turbidity_aster is a command-line program for creating relative turbidity images from ASTER 1 and 2. See *Exercises_ASTER.pdf* for details.

3.2.12 Processing Level-2 ocean color and SST data

This is a suite of programs to map and composite MODIS-Aqua, MODIS-Terra, SeaWiFS, MERIS, OCTS and other Level-2 datasets. You can use the Ocean Color "Level 1 and 2 Browser" to find and order the datasets at http://oceancolor.gsfc.nasa.gov/cgi/browse.pl?sen=am. The daily mapped composites are generated with *wam_l2_map* that uses a set of Level-2 flags to eliminate low-quality pixels. Composites from 2 sensors (e.g. MODISA and SeaWiFS or MODISA and MODIST) are composited with *wam_composite_2sensors*. Composites from 2 or

more sensors can be composited with *wam_composite_sensors*. After that 5-day composites are generated with *wam_composite_2x* using the daily composites as input. After that you can take the 5-day composites and composite further into 15-day composites with *wam_composite_2x*. Daily composites are composited into monthly composites with *wam_composite_month*. More details and examples are provided in *Exercises_SeaWiFS_Aqua_Level2.pdf* and *Exercises_SeaWiFS_Aqua_Level2_Appendix_Your_area.pdf*. A short introduction is also in section "Using MODIS-Aqua Level-2 images" of *Exercises_WIM_WAM.pdf*.

Starting with *wam_l2_map* version 4.0 (January 2011) you can select variables other than Chl-a (including MERIS *algal_1* and *algal_2* products) and SST. The selected variables can be specified with the var=VARIABLE option, e.g. var=nflh (where *nflh* is the natural fluorescence line height. The scaling standard output variables is *Pathfinder-SST* for SST and *Log-Chl* for Chl-a. For selected variables the output of the mapped data file corresponds to the scaling of the Overlay file that typically has the color scale or no scaling with *float32* input. The mapped *float32* datasets may not be ideal for visualization and you may want to find a better scaling for other variables using batch conversion with *wam_series*. Note that you can also select Chl-a datasets with the *var=chl* option and the difference with the default option is that the mapped dataset will have the scaling of the overlay file (or *float32* if Overlay file not given) but will always be in the 1-byte-per-pixel *Log-Chl* scaling with the default option.

Starting with *wam_l2_map* version 4.14 (December 2013) the default SST variable includes SST4, i.e. the 4 micrometer SST available from daytime MODIS images. If both standard SST (11 micrometer) and SST4 files are in the same folder then both are read and merged into a common daily mapped SST image. As SST and SST4 have a significant bias, it is recommended to keep SST and SST4 files separate and not to merge them.

wam_l2_map

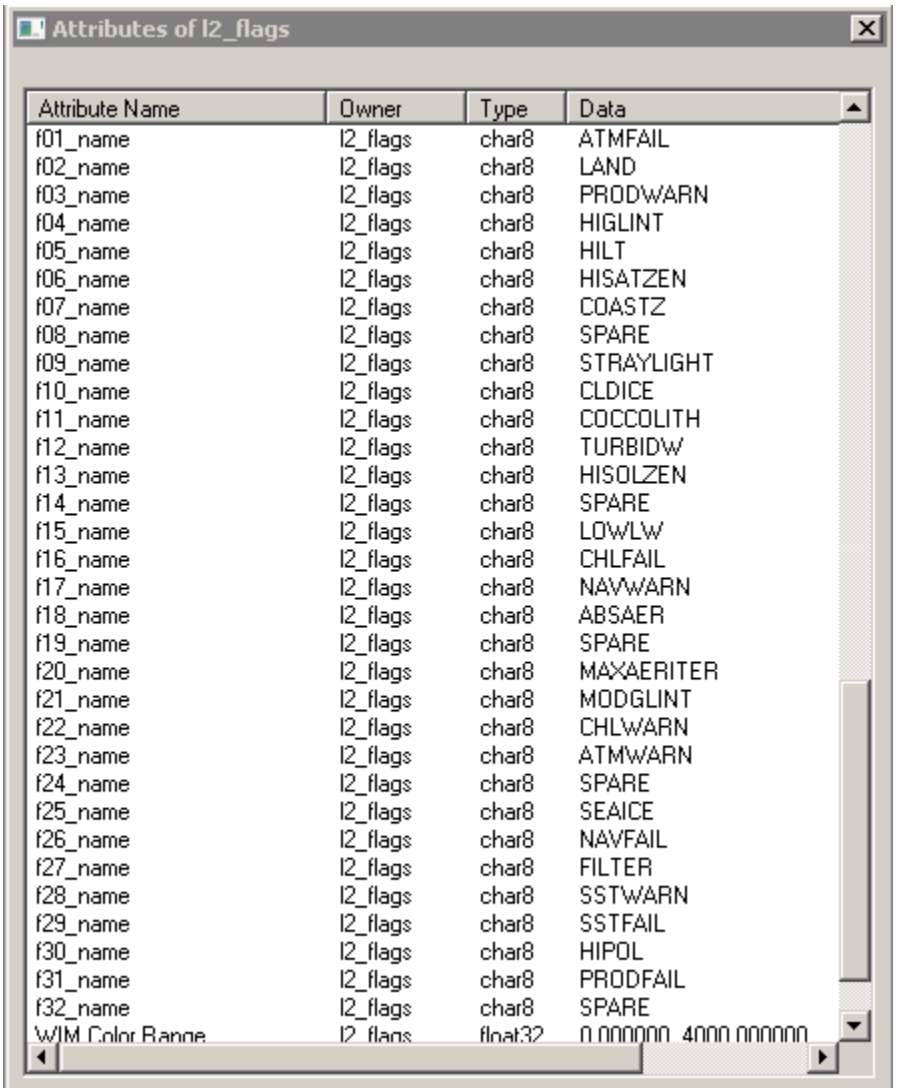
```
wam_l2_map vers. 4.19
Usage1:  wam_l2_map Pattern TargetFile [OverlayFile [xPos yPos]]
  or
Usage2:  wam_l2_map Pattern TargetFile [xPos yPos]
Pattern is a pattern for matching HDF4/netCDF filenames
TargetFile is a HDF4 file with the target projection
Optional OverLayFile is a HDF4 file for annotated images.
The reason for Usage2, i.e. a separate Target and Overlay files is
 that Overlay can be smaller than the target projection.
For large images it may be appropriate to make the annotated images smaller.
Annotation is written into the image at xPos of x-position
  and yPos of y-position. Negative vakues mean no annotation.
Note: Output is NOT to the current folder but to the directory one level up
 from the data files!
! Do not mix different years and different types of files, e.g. OC and SST!
The color range is taken from the OverlayFile or Target map or set directly, e.g.
 ColorMin=0.01 ColorMax=1
Palette file can be specified by lut=LUTFILE.lut in the command line,   e.g.
lut=chl2_white_end.lut
The default variables are Chl and SST, other variables can be specified with
  var=VARIABLE  where VARIABLE is the variable name, e.g. var=kd_490

Advanced options, mostly for testing or special cases:
```

noflags = do not use L2 flags for screening bad Chl pixels
nocloud = do not use declouding (expanding the cloud mask) for Chl
maxSSTQuality? specifies the maximum valid SSTflag quality, e.g.
maxSSTQuality1 keeps quality values 0 and 1;
the default is maxSSTQuality0, i.e. to keep only pixels with quality 0
algal_2 = use algal_2 datasets of MERIS for Chl
flags=12345678901..... with 0 for the flag to be ignored and any other
character for the flag to be used. There are 32 flags!
To see the standard flags type: wam_l2_map Pattern Target flags=

The following L2 flags (if set) make a pixel invalid when using the default option:

ATMFAIL, LAND, BADANC, HIGLINT, HILT, HISATZEN, CLDICE, HISOLZEN, LOWLW, CHLFAIL, CHLWARN, SEAICE, NAVFAIL.

With the `noflags` option the L2 flags are not used and the pixel value is used and checked for validity by its range (e.g. negative or 0 values are invalid). While some of the L2 flags represent critical failure of the retrieval (e.g. LAND, CLDICE, SEAICE), others just indicate questionable quality of the retrieval and still produce a L2 output value. You can see the names of all L2 flags by loading the l2_flags dataset to WIM and looking at the Attributes. They look like that:



When you are specifying the flags to be used to validate pixels you only need to specify until the last flag that has 1 (i.e. is used to discard a pixel) as the following (unspecified) flags are assumed to have no influence (i.e. are 0 in the flags sequence).

The default SST quality level is 0, i.e. only the maximum quality is accepted. Using only pixels with the maximum SST quality eliminates also all pixels in high gradient regions, e.g. fronts that are considered questionable by default but are usually good front pixels and will be eliminated. Therefore, if you want to include frontal pixels, you need to allow lower quality pixels.

The following are some examples of using *wam_l2_map*:

Standard Chl-a:

wam_l2_map 2012\A2012* Map.hdf Overlay.hdf 382 19 lut=chl1_white_end.lut

Standard SST:

wam_l2_map 2012_SST\A2* Map.hdf Overlay.hdf 382 19 lut=chl2_white_end.lut

SST with maxSSTQuality2:

wam_l2_map 2012_SST\A2* Map.hdf Overlay.hdf 382 19 lut=chl2_white_end.lut maxSSTQuality2

Selected Chl-a that keeps the original scaling for the mapped dataset:

wam_l2_map 2012\A2012* Map.hdf Overlay.hdf 382 19 lut=chl1_white_end.lut var=chl

Selected *nflh* that uses Log scaling for the annotated and PNG files:

wam_l2_map 2012\A2012* Map.hdf Overlay.hdf 382 19 lut=chl1_white_end.lut var=nflh

Selected *kd_490* that uses Log scaling for the annotated and PNG files:

wam_l2_map 2012\A2012*.hdf Map.hdf Overlay.hdf 382 13 var=Kd_490 lut=chl1_white_end.lut

Selected *Rrs_443* that uses Log scaling for the annotated and PNG files:

wam_l2_map 2012\A2012*.hdf Map.hdf Over.hdf 382 13 var=Rrs_443 lut=chl1_white_end.lut

Selected *Rrs_443* while using the first 4 flags only (ignoring the other flags):

wam_l2_map 2012\A2012*.hdf Map.hdf Over.hdf 382 13 var=Rrs_443 flags=1111

Note that while the *Map.hdf* file can be the same for each variable, the *Overlay.hdf* files are typically different as they include the specific color bar with a specific scaling and range of values. The scaling information for the annotated images, such as pixel type, scaling type, slope, intercept, color stretch *Start* and *End* values are taken from the Overlay file. You can also convert and annotate the images later with *wam_series*.

wam_composite_2sensors

```
wam_composite_2sensors  vers. 2.50
Usage: wam_composite_2sensors Pattern1 Pattern2
If images are of different size then Pattern2 will be remapped to Pattern1
Optional arguments:
Overlay=File specifies overlay image in HDF4, e.g. with color scale, etc.
xPos=x is annotation's x position, yPos=y is annotation's y position,
PriorityOne=yes means that Sensor1 has priority and the composite is made of
  Sensor1 pixels first and, if missing, Sensor2 pixels. In default case pixels
  are the average of the valid pixels of both sensors 1 and 2.
Sds1Name=Name1 reads the Name1 dataset from files of Pattern1,
  default is sds number 0
Sds2Name=Name2 reads the Name2 dataset from files of Pattern2,
  default is sds number 0
Sds1=M reads the Mth dataset from files of Pattern1, default is 0
Sds2=N reads the Nth dataset from files of Pattern2, default is 0
validMin1=X and validMax1=Y specify valid range for dataset 1, default is to detect
validMin2=Z and validMax2=W specify valid range for dataset 2, default is to detect
lut=LUTfile  where LUTfile is a palette file, e.g. chl2_white_end.lut
CMin=XX and CMax=YY specify the color stretch Start and End values
  If CMin and CMax are not specified, they will be taken from the OverlayFile!
```

wam_composite_sensors

```
wam_composite_sensors  vers. 1.0
Usage: wam_composite_sensors Pattern1 Pattern2 ... PatternN
If images are of different size then all will be remapped to size of Pattern1
Optional arguments:
validMin=X and validMax=Y specify valid range, default is to detect
scaling=Lin converts the default float32 image to byte with linear scaling
  with slope=X and intercept=Y, default slope=1 and intercept=0
scaling=Log converts the default float32 image to byte with log scaling
scaling=LogChl converts the default float32 image to byte with log scaling
  with slope=0.015 and intercept=-2
lut=LUTfile  where LUTfile is a palette file, e.g. chl2_white_end.lut!
```

wam_composite_sensors is similar to *wam_composite_2sensors* but allows compositing of ANY number of datasets and not just 2. It has a few other differences with the previous one, therefore, *wam_composite_2sensors* is not completely obsolete. It currently cannot select specific sds dataset with multiple are present in the file and always reads the first one in a file. It allows to specify the output scaling that *wam_somposite_2sensors* does not. For example, you can composite Chla from 3 sensors:

```
wam_composite_sensors MODISA\A2003*.nc MODIST\ T2003*9km SEAWIFS\ \S2003*.nc
```

Note that data of some sensors is in HDF4 (*9km) and some are in netCDF (*.nc).This will produce a float32 image file. If you add *Scaling=LogChl* to this command, the output will be in LogChl scaling and is better to visualize and takes less space. You can also merge *PAR* data from these 3 sensors and use *Scaling=Lin slope=0.5* options to make a composite that does not lose any significant accuracy but is smaller and can be better visualized.

wam_mosaic_2sets

```
wam_mosaic_2sets  vers. 1.0
Usage: wam_composite_2sensors Pattern1 Pattern2
```

This is similar to *wam_composite_2sensors* but instead of using the average of valid corresponding of 2 images, it uses the sum. Imagine that you have 2 sets of images: one set of the northern hemisphere and another set of the southern hemisphere. Both sets are empty in the opposite side of the earth. If you make the usual composite then you would get half of the real value if empty is counted as zero. When using *wam_mosaic_2sets* the corresponding images are found by their date and the sum of the corresponding pixel values is used for the resulting image.

wam_composite_2x

```
wam_composite_2x vers. 3.19
Usage: wam_composite_2x Pattern Days [Overlay [xPos yPos [LutFile [ColorMin Colo
rMax]]]]
Pattern is a matching filename pattern
Days is the range of days to be composited
Overlay is an HDF4 file for the overlay; use dummy name if no overlay
xPos is annotation x position, yPos is annotation y position; use negative for
  no annotation
LutFile is external LUT file,
ColorMin and ColorMax are respectively the pixel values of color stretching
```

Optional arguments:
count=YES saves the count of valid pixels
name=NAME selects specific dataset by name (instead of the first)
sds=INDEX selects a specific dataset by index (default is sds=0)
validMin=X and validMax=Y force the use of those values instead of the
automatically detected values
Note: if you mix years then you will have monthly mean over different years!
Examples:
wam_composite_2x D:\Sat\2012\A*.hdf 5
wam_composite_2x D:\Sat*.nc 5 sds=3

wam_composite_intime

wam_composite_intime vers. 2.1
Usage: wam_composite_intime Pattern StartYear StartDay EndYear EndDay [OverlayFi
le [xPos yPos [LutFile [Min Max]]]]
Pattern is a matching filename pattern,
StartYear, StartDay is the start of the period of compositing,
EndYear, EndDay is the end of the period of compositing,
OverlayFile is an overlay (HDF) file for the quick-look overlay
xPos is annotation x position, yPos is annotation y position
LutFile is external LUT file, Min and Max are respectively the pixel values of color stretching

wam_composite_month

wam_composite_month vers. 1.22
Usage: wam_composite_month Pattern [Overlay [xPos yPos [LutFile [Min Max]]]]
Pattern is the matching filename pattern
Overlay is the optional overlay file (must be of the same size!)
xPos is annotation x position, yPos is annotation y position
LutFile is external LUT file
Min and Max are respectively the pixel values of color stretching
Optional arguments:
count=YES saves the count of valid pixels
name=NAME selects specific dataset by name (instead of the first)
sds=INDEX selects a specific dataset by index (default is sds=0)
validMin=X and validMax=Y force the use of those values instead of the
automatically detected values
Note: if you mix years then you will have monthly mean over different years!
Examples:
wam_composite_month D:\Sat\2012\A*.hdf
wam_composite_month D:\Sat*.nc sds=3 count=yes

This is a powerful command to create monthly composites from series of daily datasets. The dates are automatically read from the file and the composites are made for each month. Typical usage is in creating monthly time series from daily datasets as shown in multiple exercises, e.g. *Exercises_SeaWiFS_Aqua_Level2.pdf*. The files to be composited may have more than one dataset and in that case you can pick which dataset is to be composited. For example, if the individual HDF4 files have the following datasets: *a490*, *adg443*, *aph443*, *bbp490*, *Chl-a_443*,

l2_flags_QAA then you can make monthly composites of only the *Chl-a_443* datasets with a command like this:

```
wam_composite_month  1997\S1997*.hdf   name=chl
```

Note that the argument name is case-insensitive and only the first matching string is needed to specify the dataset (*chl* instead of *Chl-a_443*). The output files will be named like *S1997335.S1997365_Chl-a_443_comp.hdf* or like *S1997335_S1997365_chl_comp.hdf* in case of standard variables. Note that the naming convention is not standard and may not be optimal in every case.

wam_composite_year

```
wam_composite_year vers. 1.1
Usage: wam_composite_year Pattern [Options]
  where Pattern is matching filename pattern
Optional arguments:
Overlay=overlay is a map or overlay file for remapping
xPos=X is annotation x position, yPos=Y is annotation y position
Lut=lutFile is external LUT file,
Min=X and Max=Y are the color min and max for float32 images
Sum=yes sums the pixel values instead of compositing
The following example creates sums of monthly image values  for each year:
wam_composite_year H:\L3\ Monthly\ A*.hdf H:\Projections\map.hdf sum=yes
```

wam_composite_last

```
wam_composite_last vers. 1.5
Usage: wam_composite_last Pattern [NLastImages [OverlayFile [xPos yPos [lutFile
[Min Max]]]]]
   Pattern is file name pattern;
   NLastImages is the Number of last images to composite, 5 is default;
   OverlayFile is the overlay image in HDF file;
   xPos is annotation x position, yPos is annotation y position
   lutFile is external LUT file, Min and Max are respectively the pixel values of color stretching
Note: Don't mix different years as the year number is ignored!
```

wam_composite_list

```
wam_composite_list vers. 1.10
Usage: wam_composite_list List [OverlayFile [xPos yPos [LutFile [Min Max [Cruise]]]]]
   OverlayFile is optional overlay as HDF
   xPos is annotation x position, yPos is annotation y position
   negative xPos or yPos means NO label!
   LutFile is external LUT file, Min and Max are respectively the pixel values of color stretching
   Cruise is an optional cruise label, e.g. CCE-LTER
```

wam_composite_running

```
wam_composite_running vers. 1.2
Usage: wam_composite_running Ndays Pattern [OverlayFile [xPos yPos [LutFile [Min Max]]]]
```

Ndays/2 is the number of days before and after to make the average.
Note that Ndays=10 makes running averages for 5 days before and 5 days after. With daily images that actually uses 11 images in total, assuming that all daily images exist.
xPos is annotation x position, yPos is annotation y position
LutFile is external LUT file,Min and Max are respectively the pixel values of color stretching

wam_composite_running is easier to understand with an example. In a typical case daily images are missing large areas due to clouds. We therefore want to use 10-day average images centered at the current day. This is called running averaging. For each day we average 5 days before and 5 days after the current day, that makes an 11 day composite. Of course, this assumes that we have all daily images. In the beginning and end of the series we don't have al the required images either before the current day or after the current day. Therefore the beginning and end of the series will have incomplete composites, i.e. over less than 11 days. You can choose any integer number of days for the length of compositing (Ndays). That integer will be divided by 2 and all images from center day - Ndays/2 to center day + Ndays/2 will be composited. The output images are written one level up in the directory tree. For example, let's run the following command:

wam_composite_running 10 chl_day\C*mapped.hdf
F:\Cal\cal_aco_3840_red8_stas_chl_47_200.hdf 186 13 chl1_white_end.lut 48 200

Here we use 10-day running composites, daily mapped files in the *chl_day* directory, a Chl overlay file with *color scale* `cal_aco_3840_red8_stas_chl_47_200.hdf`, we write the annotation at x=186 and y=13, use LUT files *chl1_white_end.lut* and stretch colors from 48 to 200 (corresponding to Chl-a concentration of 0.05 to 10 mg m^{-3}). Please note that we use only C*mapped.hdf files in the compositing and do not use the annotated *.*png* and *_*comp_annot.hdf* files. We produce three files per composite, something like that:

C2006360_C2006365_chl_comp.hdf

C2006360_C2006365_chl_comp.png

C2006360_C2006365_chl_comp_annot.hdf

. . . . . .

C2006365_C2007010_chl_comp.hdf

C2006365_C2007010_chl_comp.png

C2006365_C2007010_chl_comp_annot.hdf

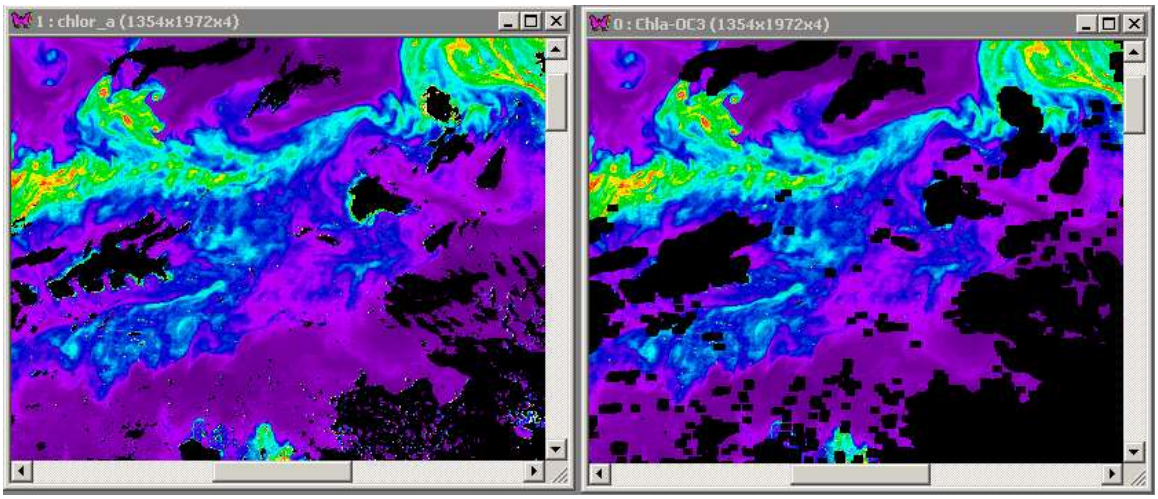
As you can see, the compositing interval extends from 2006 to 2007. For the very first composite (day 360) the preceding images are missing and only the following 5 images are used. The last composite covers 11 images, i.e. 5 images before and 5 images after the central image.

wam_rrs_l2

wam_rrs_l2 vers. 1.2
Usage: wam_rrs_l2 Pattern Algorithm [TargetFile][Noflags]
Pattern is a matching set of HDF files with Rrs
Algorithm can be: OC2, OC3, OC4, OC3L, OC4L, CHLCALFIT3, CHLCALFIT4,
CHLSPGANT3, CHLSPGANT3BLENDED, CHLSPGANT4,CHLSPGANT4BLENDED
CHLCALFIT3 and CHLCALFIT4 are OC3/OC4 type fits to California Current Chl;
CHLSPGANT3 and CHLSPGANT4 are Southern Ocean Chl-a algorithms with

the respective BLENDED versions blend with NASA standard OC3/OC4;
OC3L and OC4L are Glenn Cota Arctic fits.
Optional TargetFile is a HDF file with a target projection
With optional argument Noflags will not use L2-flags
Example:
wam_rrs_l2 A2004\Good*.hdf SPGANT3 noflags

wam_rrs_l2 applies various algorithms to Rrs bands in standard ocean color Level-2 files and saves the calculated product in a HDF file with latitude/longitude arrays. Standard L2-files have the standard *chlor_a* product that is calculated, for example, using OC3 for MODIS and OC4 for SeaWiFS. This command allows applying other algorithms. Currently only various Chla products are implemented. The command also performs masking depending on L2-flags and expands the masked area to get rid of pixel noise ("speckle"). You can also calculate the standard Chla that should be very close to the *chlor_a* calculated by NASA. The following example shows the standard *chlor_a* (left image) and the calculated Chla which has blocked out a lot of the speckle but also around clouds and land.



3.2.13 Merging Level-3 data

wam_merge_l3

wam_merge_l3 vers. 1.1
Usage: wam_merge_l3 Pattern1 Pattern2 [DeltaDays [PriorityOne]]
DeltaDays (default = 2) is the time difference allowed for merging.
The best matching Image2 is used, if the time difference is < DeltaDays
In default case (no Priority) the merged pixels are the average
of the valid pixels of both Image1 and Image2 pixels.
Any argument in PriorityOne means that Image1 has priority and the
merger is made of Image1 pixels first and, if missing, Image2 pixels.
Image1 projection will be used for the merger.

wam_merge_l3 is a command-line program for merging different Level-3 datasets. An obvious application is to merge the corresponding SST fields from multiple sensors, e.g. from MODIS-Terra and MODIS-Aqua. In this case both datasets have the same quality and resolution and you

can skip the *DeltaDays* and *PriorityOne* options. Only the corresponding datasets will be merged. Merging these datasets reduces the amount of missing data due to clouds. Another, more complex example is to merge datasets with different resolutions, e.g. the all-weather but low-resolution microwave SST with the high-resolution but clear-sky infrared SST. The merged SST will have the high resolution of the infrared data (where possible) and the all-weather coverage of the microwave data where the high-resolution data is missing. A more sophisticated example of this process is the New Generation Sea Surface Temperature (NGSST) project (http://www.ocean.caos.tohoku.ac.jp/~merge/sstbinary/actvalbm.cgi?eng=1) that creates merged SST for the ocean around Japan assembled from various sources by the Kawamura Lab in the Tohoku University, Japan. Please see *Exercises_Merging_SST.pdf* details.

3.2.14 Inherent Optical Properties (IOP)

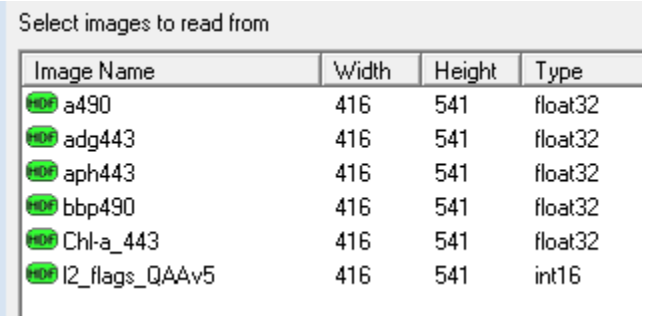
wam_qaa_l2

```
wam_qaa_l2 vers. 4.1
Usage:  wam_qaa_l2 Pattern [TargetFile][Chl][Noflags][CalFit]
Pattern is a matching set of HDF files with Rrs
Optional TargetFile is a HDF file with a target projection
Output is currently fixed to: a490, adg440, aph440, bbp490, flags;
  other wavelengths (e.g. 412, 443, 490) of a, aph, adg, bbp can be added
  if needed but only after modifying the code
With optional argument Chl it will calculate Chl-a from aph440
With optional argument Noflags it will not use L2-flags
CalFit forces to use coefficients fitted to CC matchups
```

This command will read standard NASA Level-2 ocean color files with bands of remote sensing reflectance (*Rrs*) and will apply Lee's QAA version 5/6 algorithm to calculate inherent optical properties (IOPs) such as total absorption at 490 nm (*a490*), phytoplankton absorption coefficient at 440 nm (*aph440*), detrital and gelbstoff absorption coefficient at 440 nm (*adg440*), particulate coefficient of backscatter at 490 nm (*bbp490*). Output is saved in a single HDF4 file. If a target projection file is given then all images will be remapped to that projection. If an argument "chl" is given then Chl-a concentration will be calculated from *aph440* using the Bricaud et al (1998) model. If "noflags" argument is given then the L2-flags are not used. Normally the L2-flags are used to eliminate bad and questionable pixels. Cloud edges are extended (dilated) in order to remove the often bad pixels next to clouds. The same L2-flags and procedures are used in *wam_l2_map*. For example, the following command will process all matching SeaWiFS L2 files in a folder "0", remap to the projection in *Map.hdf* and add Chl-a:

```
wam_qaa_l2  0\S1999*.L2_MLAC_OC.x.hdf  Map.hdf  chl
```

A sample output file will have the following datasets:



This command has been adjusted for bands on multiple sensors. Currently this command has been adjusted to run on the following sensors: SeaWiFS, MODIS and MERIS. MERIS has different L2-flags that are being used. The following command will process MERIS L2-files (RR files) and map to the projection in *Map.hdf* and add Chl-a:
wam_qaa_l2 MERIS2011\ MER_RR_*.hdf E:\Projections\Map.hdf chl

MERIS L2 files have been converted from *.N1 format to HDF4 format with *wam_convert_n1*. Please note that *wam_qaa_l2* is rather slow as it is running pixel-wise through images. The Chl-a_443 dataset is in *Float32* format and can be converted to scaled byte format with *Log-Chl* scaling in WIM or using *wam_series*.
Note that the output wavelengths in *wam_qaa_l2* are selected for the input of the absorption-based productivity (Aph-PP) algorithm in wam_npp_lee.

wam_qaa_l3

wam_qaa_l3 vers. 4.2
Usage: wam_qaa_l3 Rrs5XX_pattern [aXXX aphXXX adgXXX bbpXXX chl CalFit]

Rrs5XX_pattern is the filename pattern for Rrs of the green band, e.g.
SeaWiFS Rrs_555 or MODIS, OCTS, MERIS equivalent Rrs files
It is assumed that all Rrs files are in the same folder.
Optional arguments select which products to output;
the XXX in each variable specifies the wavelength.
Wavelengths are limited to the wavelengths used in QAA.
Not all wavelengths are available for output.
Currently only wavelengths 440 and 490 nm have been implemented
Default is to output a490, aph440, adg440, bbp490 and QAA-flags.
The following command will output a490, aph440, bbp490, Chl and flags:
wam_qaa_l3 2002\A2002001*Rrs_547_9km a490 aph440 bbp490 Chl
Input for the Lee NPP model is: a490, aph440, bbp490 (no separating commas!)
For large images (e.g. 8640 x 4320) you will probably run out of memory and
need to reduce the number of output products in a single run.
CalFit forces to use coefficients fitted to CC matchups.

This command will read standard NASA Level-3 ocean color files with remote sensing reflectances (*Rrs*) and will apply the Lee et al. QAA version 5/6 algorithm to calculate inherent optical properties (IOPs) such as total absorption coefficient at 490 nm (*a490*, m^{-1}), phytoplankton coefficient of absorption at 440 nm (*aph440*, m^{-1}), detrital and gelbstoff coefficient of absorption at 440 nm (*adg440*, m^{-1}), particulate coefficient of backscatter at 490 nm (*bbp490*, m^{-1}) and optionally the Chl-a concentration calculated from *aph440* using the

Bricaud et al (1998) model. It is assumed that all files with the required *Rrs* bands are in the same folder. You can choose the output variables with an argument string *aXXX* (total absorption), *aphXXX* (phytoplankton absorption), *bbpXXX* (particulate backscatter) where "*XXX*" is the band wavelength in nm. Optional outputs are also Chl and Flags (both without the specific band wavelength). Default output, i.e. without any optional arguments, is *aph440*, *adg440*, *bbp440* and *flags* but not Chl-a. This command is fast (e.g. compared to the related command *wam_qaa_l2*) as it works with floating point arrays and not with image structures, however, it will run out of memory when multiple variables are required at the global 4-km resolution (8640 x 4320 pixels). At 9-km resolution (4320 x 2160 pixels) it should have no problems. When it runs out of memory you need to reduce the number of output variables for a run. Currently only bands 440 nm and 490 nm are implemented for output. For example, the following command will output *a490*, *aph440*, *bbp490* and flags. The first 3 are the inputs for the Lee et al. (2011) primary production model.

```
wam_qaa_l3   Rrs_9\O1996*.L3m_DAY_RRS_Rrs_565_9km a490 aph440 bbp490 flags
```

wam_qaa_match

```
wam_qaa_match vers. 1.0
Usage: wam_qaa_match MatchFile [aXXX aphXXX adgXXX bbpXXX Flags CalFit]
  MatchFile is a CSV file with RrsXXX,... needed in QAA for that sensor
Default algorithm is QAAv5, can be switched to CalFit
CalFit forces to use QAA coefficients fitted to CC matchups
Optional arguments select which products to output;
  the XXX in each variable specifies the wavelength.
Wavelengths are limited to the wavelengths used in QAA.
Not all wavelengths are available for output.
Currently only wavelengths 410, 440 and 490 nm have been implemented
Default is to output a490, bbp490, aph440, adg440 and QAA-flags.
The following command will output a490, aph440, adg410:
   wam_qaa_match Match.csv a490 aph440 adg410
```

This command will read a CSV file (text file) with columns as in a standard WIM/WAM "match" file, i.e. Longitude, Latitide, Date, Time, …. Required columns are remote sensing reflectances (*Rrs*). These are typically extracted from L3 satellite images with *wam_match_multiband*. It detects the satellite sensor (e.g. SeaWiFS, MODIS, MERIS) and applies a sensor-sepcific version of the Lee et al. QAA version 5b algorithm to calculate inherent optical properties (IOPs) such as total absorption coefficient at 490 nm (*a490*, m^{-1}), phytoplankton coefficient of absorption at 440 nm (*aph440*, m^{-1}), detrital and gelbstoff coefficient of absorption at 440 nm (*adg44*0, m^{-1}), particulate coefficient of backscatter at 490 nm (*bbp490*, m^{-1}). You can choose the output variables with an argument string *aXXX* (total absorption), *aphXXX* (phytoplankton absorption), *bbpXXX* (particulate backscatter) where "*XXX*" is the band wavelength in nm. Additional output is QAA Flags (with the *Flags* option). Default output, i.e. without any optional arguments, is *a490*, *bbp490*, *aph440*, *adg440* and *flags*. Option CalFit switches coefficients from standard QAA to the *CalFit* version (Kahru et al., 2013).

3.2.15 Correlation between images and a point time series

wam_correlation

```
wam_correlation vers. 1.13
Usage: wam_correlation Pattern TS_File [Options]
  where Pattern is a matching pattern of HDF4 or netCDF files,
TS_File is the CSV file with time series
Options:
Column=C specifies column number to be used in the TS_File, default is 1 (i.e. 2
nd)
TimeColumn=C specifies the column # of time, default is 0
LUT=XXX specifies the LUT file, default is anomaly.lut
Sig=SS where SS can be 90, 95, 99 is significance of correlation (R)
  at that % level of probability using  Fisher transformation and the
  R values not siginificantly different from 0 at that level are set to 0
If significance level is not specified, all R values are kept
Example:
 wam_correlation Images\*.hdf TimeSeries.csv 2 anomaly5.lut 95
```

wam_correlation calculates a correlation image (with each pixel representing the correlation coefficient) between a series of images and a point time series, e.g. an ENSO index. It is matching the time points in the time series with the image with nearest timing. When the significance level (90%, 95% or 99%) is specified, all correlation values below the significance level are set to 0.

wam_correlation_series

```
wam_correlation_series vers. 1.12
Usage: wam_correlation_series  Pattern1 Pattern2
  Pattern1 and Pattern2 are the matching filename patterns
Options: Sds1=SDS1 Sds2=SDS2 Lut=LUT From=FROM To=TO Signif=X Savecount=Yes
        Lag=X Median1=Yes Median2=Yes
  Sds1=N reads the Nth dataset from files of series 1, default is 0
  Sds2=M reads the Mth dataset from files of series 2, default is 0
  Lut=anomaly.lut specifies the palette file, default is anomaly.lut
  From and To limit the seasonal range to be used.
  The units can be either Months (with monthly data) or Julian days
  (if higher frequency data are used), e.g.
  From=1 To=31 will use days 1-31, i.e. January, if using data with
  higher than monthly frequency;
  From=3 To=9 will only use months 6 to 9 if using monthly data;
  From=10 To=2 will only use months 10, 11, 12, 1, 2, i.e. the winter period
  if using monthly data.
  Signif=X where X is 90, 95 or 99 sets all correlations below the
    respective significance X to 0
  Savecount=Yes will save an image with the count of pairs used for each pixel
  Lag=X will lag 1st series by X steps relative to series 2,
```

```
  Lag=-X will move 1st series forward by X steps compared to series 2
  Median1=Yes applies median filter to series 1 before the correlation analysis
  Median2=Yes applies median filter to series 2 before the correlation analysis
Examples:
  wam_correlation_series data1\*.hdf data2\*.hdf
  wam_correlation_series data1\*.hdf data2\*.hdf Lut=anomaly5.lut From=3 To=9
  wam_correlation_series data1\*.hdf data2\*.hdf significance=90
```

wam_correlation_series calculates a correlation image (with each pixel representing the correlation coefficient) between two series of images. The HDF files may have multiple datasets and you can select the specific dataset to be used with Sds1=N and Sds2=M. The default is to use the 0^{th} dataset in each series. *wam_correlation_series* finds the closest match in time in series 2 to each dataset in series 1. If images do not match exactly, certain matching errors are allowed. You can specify a *Lag*=X and series 1 will be lagged by X steps relative to series 2. A negative lag means that series 1 precedes series 2 by X steps. If the series 2 images are of different size than series 1 images then they will be remapped to the size and projection of the series 1 images. In general, all matching image pairs from both series are used. However, sometimes it is useful to use only selected image pairs, e.g. from a certain season or time period. The optional *From* and *To* arguments allow to selectively use only image pairs of a certain time period. If *From* is larger than *To* (e.g. 11 and 2) then it is assumed that we are picking the winter months from Nov to Feb (months 11, 12, 1, 2).

wam_corrmatrix – rescinded, no longer included

```
wam_CorrMatrix vers. 2.1
Usage: wam_CorrMatrix TS1_File TS2_File [LagMonths [123456789012]]
  TS1_File is the 1st set of time series
  TS2_File is the 2nd set of time series either for masks or for a Lat-Lon grid
  Default for LagMonths is 0, for example,
    -1 means matching with 1st time series of 1 month early
     1 means matching with 1st time series of 1 month later, etc
  123456789012 specifies Months to use, default is to use all months, e.g.
    110000000000 means using only Jan-Feb
    111000000111 means using only Jan-Mar and Oct-Dec
    000000000110 means using only Oct-Nov
```

wam_corrmatrix calculates a correlation matrix between two sets of time series. It is possible to select any set of months that are used or excluded.

3.2.16 Simple utilities

MakeCircles

```
MakeCircles vers. 1.4
Usage: MakeCircles ImageFile PointFile Radius(km)
  where ImageFile is a hdf filename,
  PointFile is a text file with header and Lon, Lat,...
  Date and Time can be fake, e.g. 1/1/2011,12:00
Radius(km) is the radius of the circle in km
```

This utility helps to make circular masks with a given radius for a list of points into a base image. The circles are made with pixel value starting from 1 that is increased by one for each following point. The circles should be edited with WIM functions Edit-Draw and filled with the desired pixel values, e.g. starting from 1. Remember to stretch the colors in order to see the differences between pixel values 1 and 2, for example. Those masks (e.g. areas with pixel values 1, 2, …) will be then used as masks in *wam_statist*. Imagefile must be an HDF file. PointList is a simple text file with a header and columns of longitude, latitude, date, time, variable1, variable2,… separated with a comma. The date and time name are not actually used and could be fake. Below is a sample *PointList*:

```
Lon,Lat,Date,Time,Name
-122.515,33.5,1/1/2000,12:00,CCE-1
-120.815,34.325,1/1/2000,12:00,CCE-2
```

rcrit

```
rcrit vers. 1.0
Usage: rcrit r N
 - calculates confidence limits for given r and N
```

This utility calculates the upper and lower confidence limits (at 90%, 95% and 99% probability) for a given correlation coefficient *r* and number of pairs, *N*.
For example, for a correlation coefficient r = 0.2 and N = 130,

```
 r = 0.2, N = 130
l90 = 0.0567, u90 = 0.3352
l95 = 0.0288, u95 = 0.3598
```

This shows that at N = 130 the given correlation is significant at 90% and 95% but not at 99% (as the lower limit becomes negative, it include the no correlation, *r* = 0).

wam_add_attribute - rescinded

```
Usage: wam_add_attribute PATTERN AtrName AtrValue
      where PATTERN is a matching pattern of hdf filenames
      AtrName is a global attribute name
      AtrValue is a string for attribute value
```

wam_add_attribute is a simple utility that reads a series of matching HDF files and adds a global attribute with the same string value. An example where this utility comes handy is the following. In order to correctly calculate statistics *wam_statist* needs to determine the valid *max* and *min* of each dataset. With most standard files this should be no problem but if you have a non-standard dataset then min and max values may be incorrectly determined. For example, for a simple byte dataset with no scaling value 1 is considered the valid minimum and value 0 is considered invalid. However, if those values represent counts then 0 is a valid value. To fix this problem you can use wam_add_attribute to add a new attribute valid_range and set it to 0,254. That makes 0 a valid value. To run this for all the *.hdf files in the current folder, type:

wam_add_attribute *.hdf valid_range 0,254

Note that you cannot have a space in “0,254” as that would breake the third argument.

wam_add_name

```
Usage: wam_add_name PATTERN
```

```
  where PATTERN is a matching pattern of hdf filenames
```

wam_add_name is a simple utility that reads a series of matching HDF files and renames the first dataset (SDS) in it to the name of the file itself (without the full path). The image name (SDS in HDF) may be very long or without a useful descriptor that separating images in similar files. For example, many standard products use the same SDS name, e.g. *l3m_data*. Sometimes all the files have the same image name, e.g. *Composite[1-2] Average* and you may want to rename the image names to the corresponding filename. To run it for all the *.hdf files in the current folder, type:

wam_add_name *.hdf

wam_cut

```
wam_cut  vers. 1.3
Usage: wam_cut Pattern x1 y1 x2 y2 [SDS]
where PATTERN is a matching pattern of hdf filenames,
x1 = left, y1 = top, x2 = right, y2 = bottom pixel coordinate
```

SDS = number of SDS (0-referenced)

wam_cut is a simple utility that cuts a rectangular area specified by its left, top, right and bottom pixel coordinates from a series images in matching HDF4 or netCDF files. The same functionality is built into the GUI utility wam_series but sometimes it is more convenient to run this command line utility.

wam_minmax

```
wam_minmax vers. 1.0
Usage: wam_minmax Pattern
  where Pattern is a matching pattern of HDF filenames
```

Shows the valid range for a set of matching HDF4 files.

wam_setMinMax

```
wam_setMinMax  vers. 1.0
Usage: wam_setMinMax  PATTERN  dMin  dMax
where PATTERN is a matching pattern of HDF4 or netCDF filenames
dMin and dMax are valid Min and Max, respectively
```

Sets the valid range for a set of matching HDF4 or netCDF files.

wam_proj_lin

```
wam_proj_lin vers. 1.0
Usage: wam_proj_lin PATTERN LonA LonB LatA LatB
  where PATTERN is a matching pattern of hdf filenames
  LonA is longitude intercept
  LonB is longitude slope
  LatA is latitude intercept
  LatA is latitude slope
```

Reads a set of matching HDF files, sets projection to Linear with the specified coefficient for all datasets (SDS) in each file, saves under the same name. Files should not be write-protected. A sample command:

```
wam_proj_lin *.hdf -179 0.015 85 -0.01
```

wam_show_time

wam_show_time vers. 1.0
Usage: wam_show_time Pattern
where Pattern is a matching pattern of hdf filenames

Reads a set of matching HDF files, and prints out the start year, start day, end year and end day of each file. A sample command:

E:\L3\MODISA\L3\Monthly\CHLCALFIT3_9>wam_show_time A2011*.hdf

12 matching files found in .\

A2011001.A2011031_CHLCALFIT3_comp.hdf: 2011,1 - 2011,31

A2011032.A2011059_CHLCALFIT3_comp.hdf: 2011,32 - 2011,59

A2011060.A2011090_CHLCALFIT3_comp.hdf: 2011,60 - 2011,90

A2011091.A2011120_CHLCALFIT3_comp.hdf: 2011,91 - 2011,120

3.2.17 Sample WAM programs for learning and testing

These are simple WAM programs that are provided only as source codes and not as executables. They are simple programs that are meant for learning how to write your own WAM applications. The source codes may be rather old and not the most efficient but should be easy to understand. Please note that the tasks that these samples programs can be more easily accomplished by the existing WAM applications, e.g. *wam_series*. If you are thinking of creating your own WAM applications then you should check if this has already been accomplished in another WAM application.

test_cut_hdf

test_cut_hdf is a simple WAM program that reads all the matching *.hdf files in a predefined directory, cuts a predefined rectangular part and saves the rectangle as a JPEG image in the current directory. To build it, type

c test_cut_hdf

The program assumes that you have a specific directory "C:\sat\seawifs\L3\month\1997" and some files matching a pattern "*_CHLO" in that directory. It checks if the specified directory exists and if it has any files matching the defined pattern. It stops if it does not find any files to process. If you want to read files from another directory and/or matching other patterns you have to modify the *directory* and/or the *pattern* variables. The program then reads all matching files and saves a specified rectangle (top left=618, top=614, bottom right=1130, bottom=870) in a separate JPEG file. You may also want to specify another rectangle for your files.

Of course, a program like that is very limited: the directory, file pattern and the rectangle are all fixed. However, you can easily change them by changing the source code, recompiling and then running the modified program. You can also modify it to read a file type other than HDF. In later examples we will show how we can specify various arguments at command line, so that we can

have more flexibility. A much more advanced version of this program is included in a Windows GUI program *wam_series*.

test_band_ratio

test_bandratio is a simple but a potentially useful WAM program that shows how to read a series of image pairs and perform the *Multi - Band Ratio* operation of WIM on them. Band ratio algorithms are common in ocean color. Those algorithms are based on a ratio of two bands of the water-leaving radiances or reflectances. This example reads all the matching SeaWiFS L443 and L520 files in a predefined directory and creates the CDOM Float image of a selected rectangle using the Kahru & Mitchell (2001) empirical band ratio algorithm. Other band ratio algorithms can be easily applied. To build it, type

c test_bandratio

To run it, type:

test_bandratio

As output, the program generates a HDF file with Float values and a Byte image saved as JPEG. The conversion from Float to Byte uses Logarithmic scaling similar to the SeaWiFS Chlorophyll scaling but other scalings can be easily applied. Please note that most of the HDF file attributes are copied from the source images and may not correspond to the result image. WIM adds several attributes of its own.

wam_filter

wam_filter is a simple example how to use WAM to read a series of images, cut out a defined rectangle, and run it through a filter, e.g. a median, mean or sigma filter (please see the WIM Manual for more information), and save the filtered images. You can use *wam_series* for median filtering. The most common usage is the median filter with a kernel of 3, 5 or 7 pixels, e.g.

imgMedian = imgSource.Filter (FilterKind.Median, 5, 0.0);

FilterKind specifies the median, mean or sigma filter. The last parameter (a floating point number) is only used for the sigma filter and ignored for the others. A simplified syntax can be applied to use the median filter:

imgMedian = imgSource.FilterMedian (kernel_size);

To build it, type

c wam_filter

To run it, type:

wam_filter

As output, the program generates filtered JPEG file(s) in the current directory. The example can be easily modified or used as a building block in other applications.

wam_attribute

wam_attribute is a simple utility that reads a series of HDF files in a folder and prints out the filename and the value of a selected integer attribute (`"Start Day"`). Of course, it is just an example that you need to modify in order to apply for your task. For example, you may want to

modify the source folder where the files are expected to be, the pattern to pick the files, and the name of the attribute.

To run it, type:

wam_attribute

You can pipe the output to a text file *start_day_attribute.txt*, e.g. by typing

wam_attribute > start_day_attribute.txt

Currently this function returns correctly only an *int16* attribute and throws an exception if the specified argument does not exist. If you need to read attributes of other types (*int32*, *float32*, *text*) you need to modify the source code and recompile. More general functions returning other types of attributes may be added in the future. This function may be useful, for example to extract the "Start Day" attribute for calculating primary production (e.g. for_*wam_npp*) that needs Julian day of the middle of the period. If the images are monthly composites then the approximate middle Julian day can be calculated as the value of "Start Day" + 15.

test_statist

test_statist is a simple example showing how to use the *GetStatistics* function in WAM. For more advanced applications see *wam_statist_sta, wam_integ_mask, wam_series* and *wam_statist.*

test_statist opens a predefined image, calculates statistics using the *GetStatistics* function for a predefined rectangular area and saves the statistics in a file *statist.txt*. The syntax to use GetStatistics is:

Statistics s = img.GetStatistics (x1, y1, x2, y2, fMin, fMax);

Here x1 and y1 specify the top left corner, x2 and y2 the bottom right corner and fMin and fMax are the valid Min and Max of the geophysical values. The output of *GetStatistics* is a structure *Statistics* that has the following members: *Nin* (the number of valid pixels), *Nout* (the number of invalid values), *Min*, *Max*, *Mean*, *Median*, *StDev*.

You can modify the source code and recompile by typing:

c test_statist

To run it, type:

test_statist

wam_rgb

wam_rgb gives a simple example how to create a series of RGB (red-green-blue) composites from combinations of three image bands. This example reads all matching image files from a selected folder, calculates RGB composites from selected bands and saves the RGB image as a JPEG file.

wam_to_hdf

wam_to_hdf is an example that shows how to convert a large set of plain raster image files to HDF and in the process add many important attributes. The attributes specifying the time of the image are necessary when creating time series of images or evaluating match-ups with in situ

data. The arguments are: the name of the folder with the input files and the pattern of filenames. It is assumed that the input files are Byte images. *wam_to_hdf* reads all matching files in the specified folder, adds many attributes and saves as HDF. Attributes like “Start Year”, “Start Day”, etc. are used in time series analysis. This example can be easily modified for other related tasks.